Terascale Optimal PDE Solvers

(presentation for Salishan Conference on

"Scalability and Performance: Finding the Right Balance")

David E. Keyes

Center for Computational Science

Old Dominion University

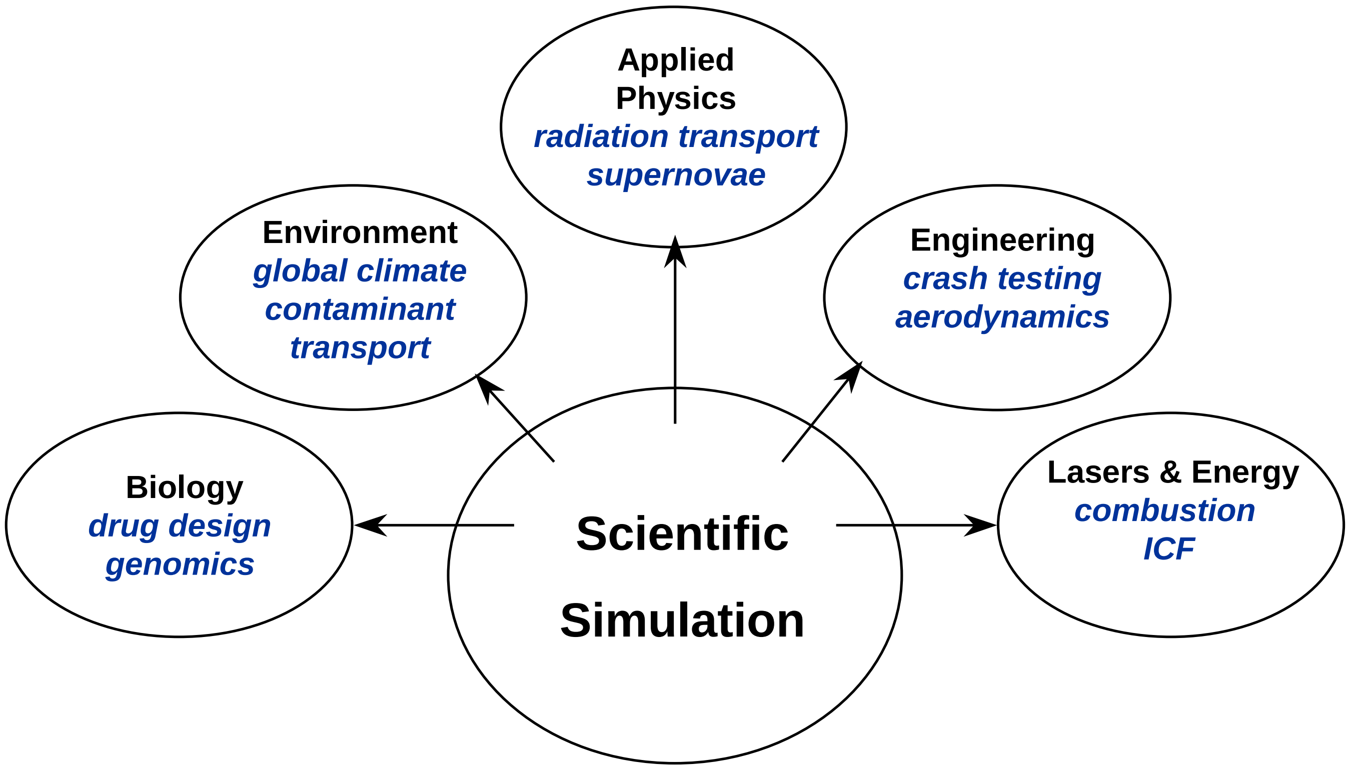
Links to active themes

- My metric: ability to *resolve all relevant scales* (and control the rest) in important apps
- My caveat: concentrating (in this talk) on just one phase of the computation – *the solver*
- My apology: assume a *reliable machine of "type C"* (à la Burton Smith) – can be relaxed, in part
- My confidence: success will *greatly expand the HPC share* of the commercial market – upward spiral effect ☺

Plan of presentation

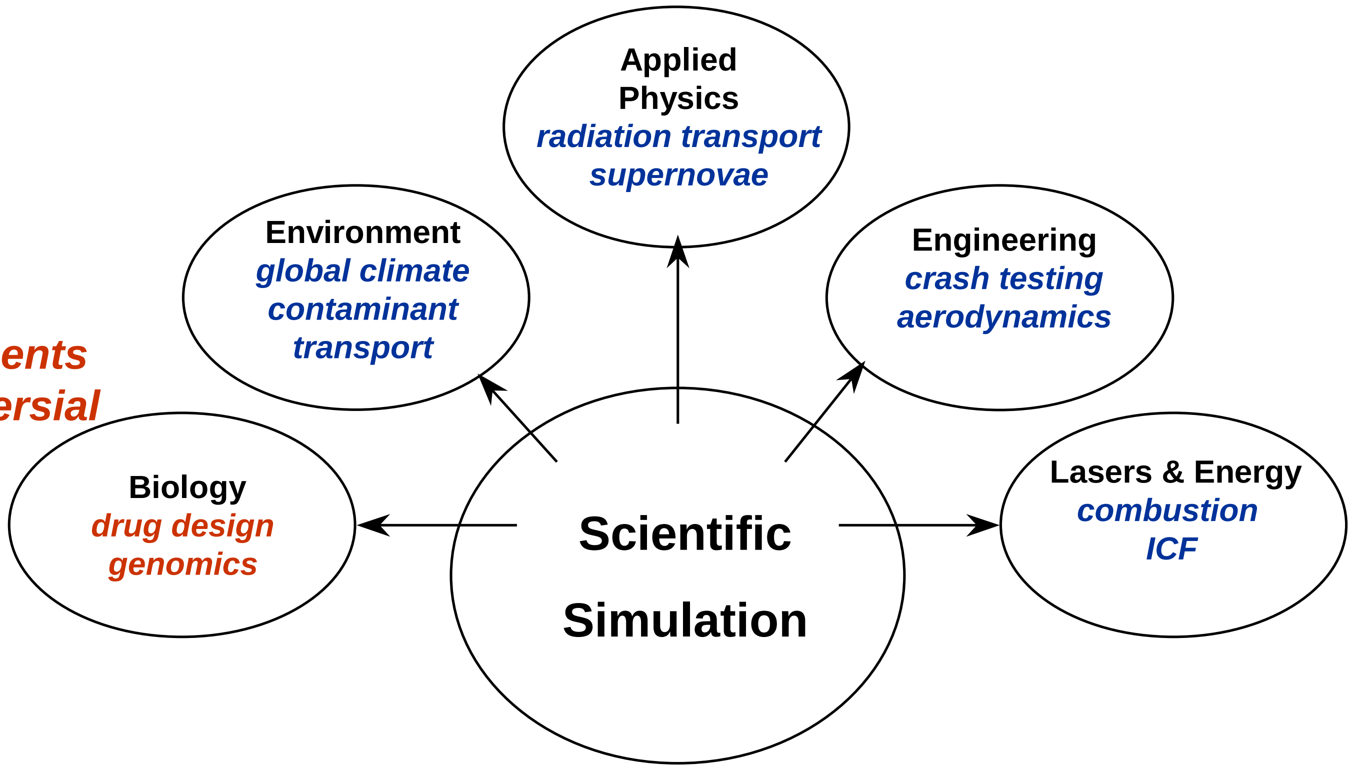
- Imperative of “optimal” algorithms for terascale computing
- Basic domain decomposition and multilevel algorithmic concepts
- Illustration of solver performance on ASCI platforms
- Terascale Optimal PDE Simulations (TOPS) SciDAC ISIC software project
- Conclusions

Terascale simulation has been “sold”



In these, and many other areas, simulation is an important complement to experiment.

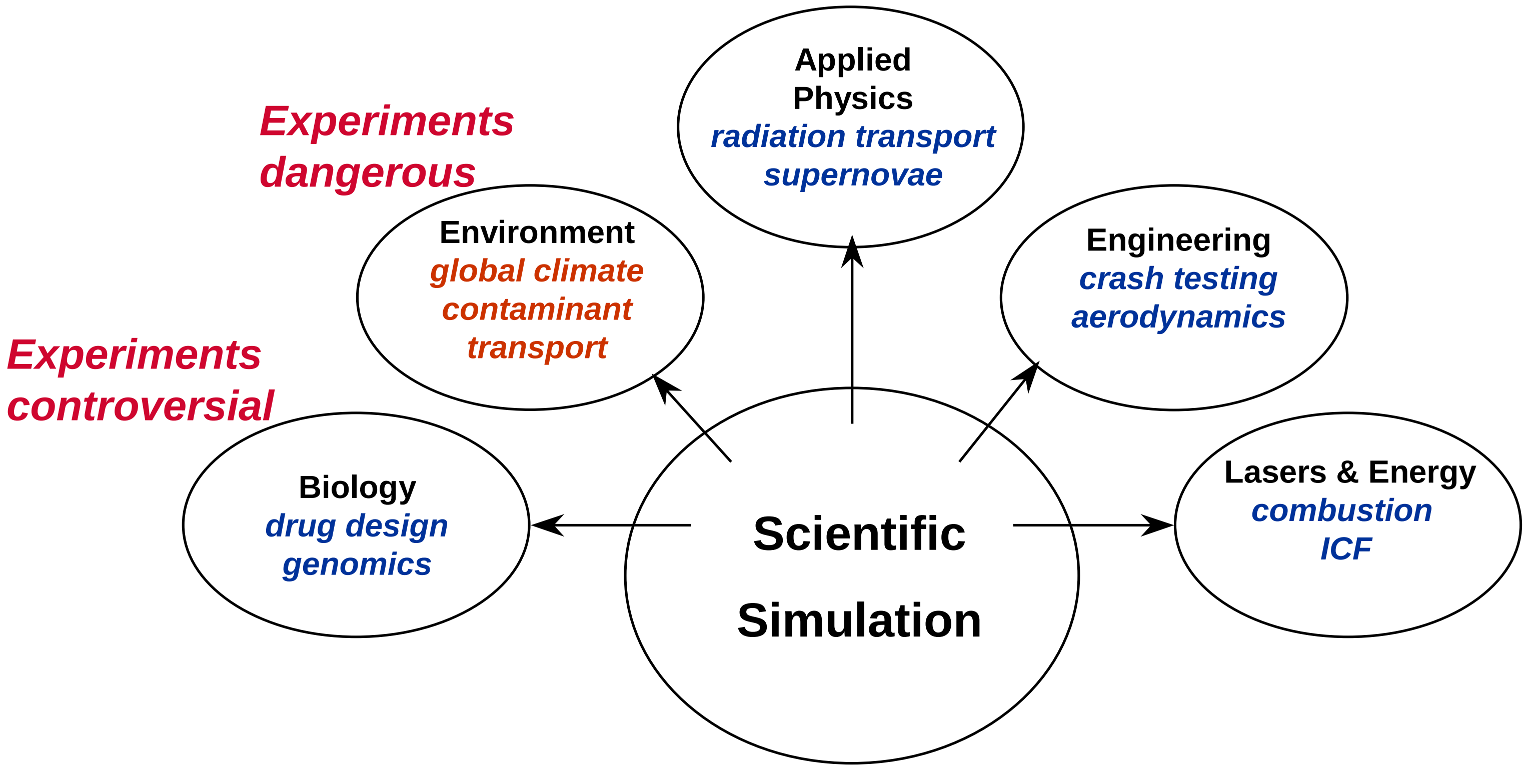
Terascale simulation has been “sold”



Experiments controversial

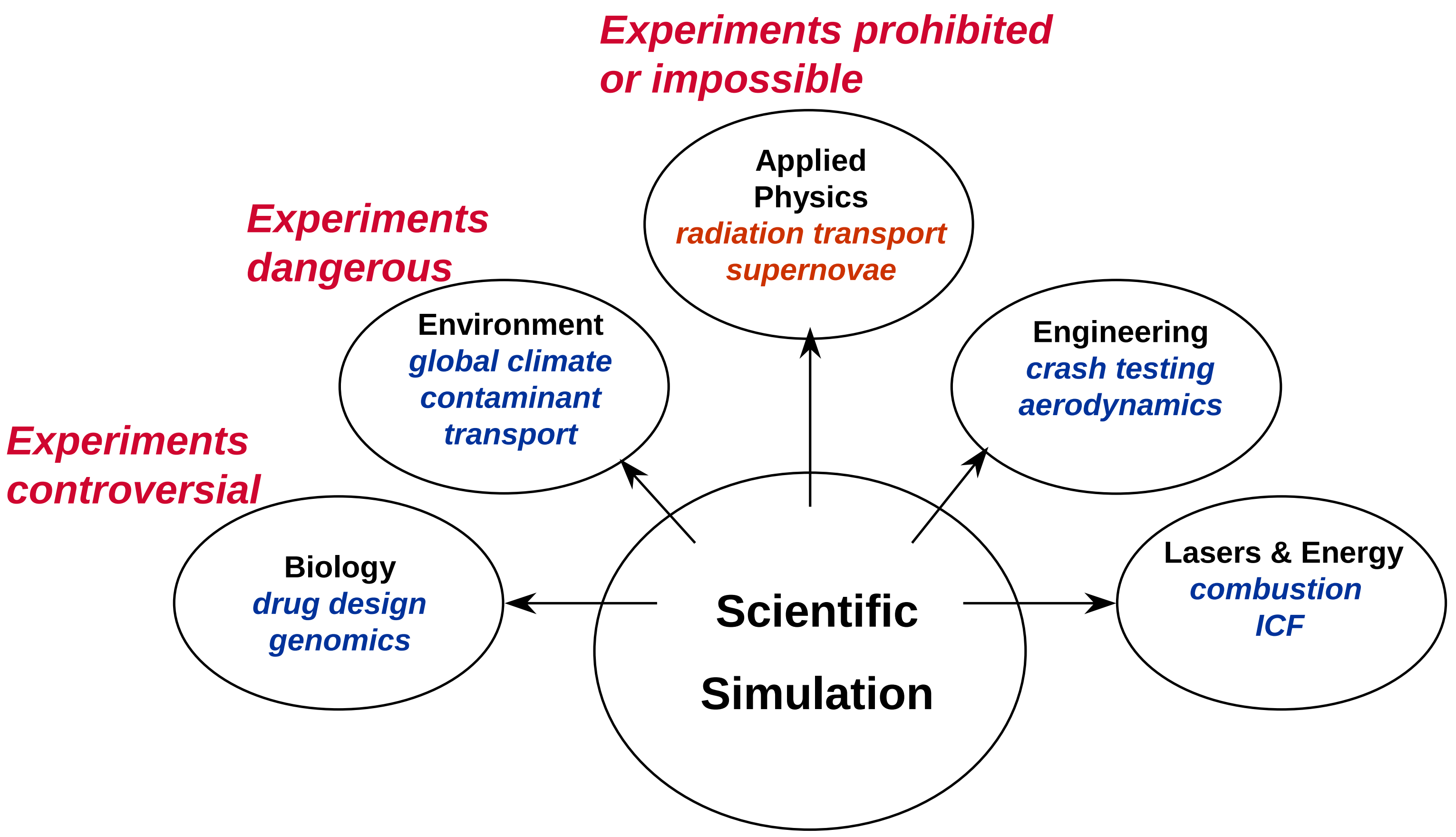
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been “sold”



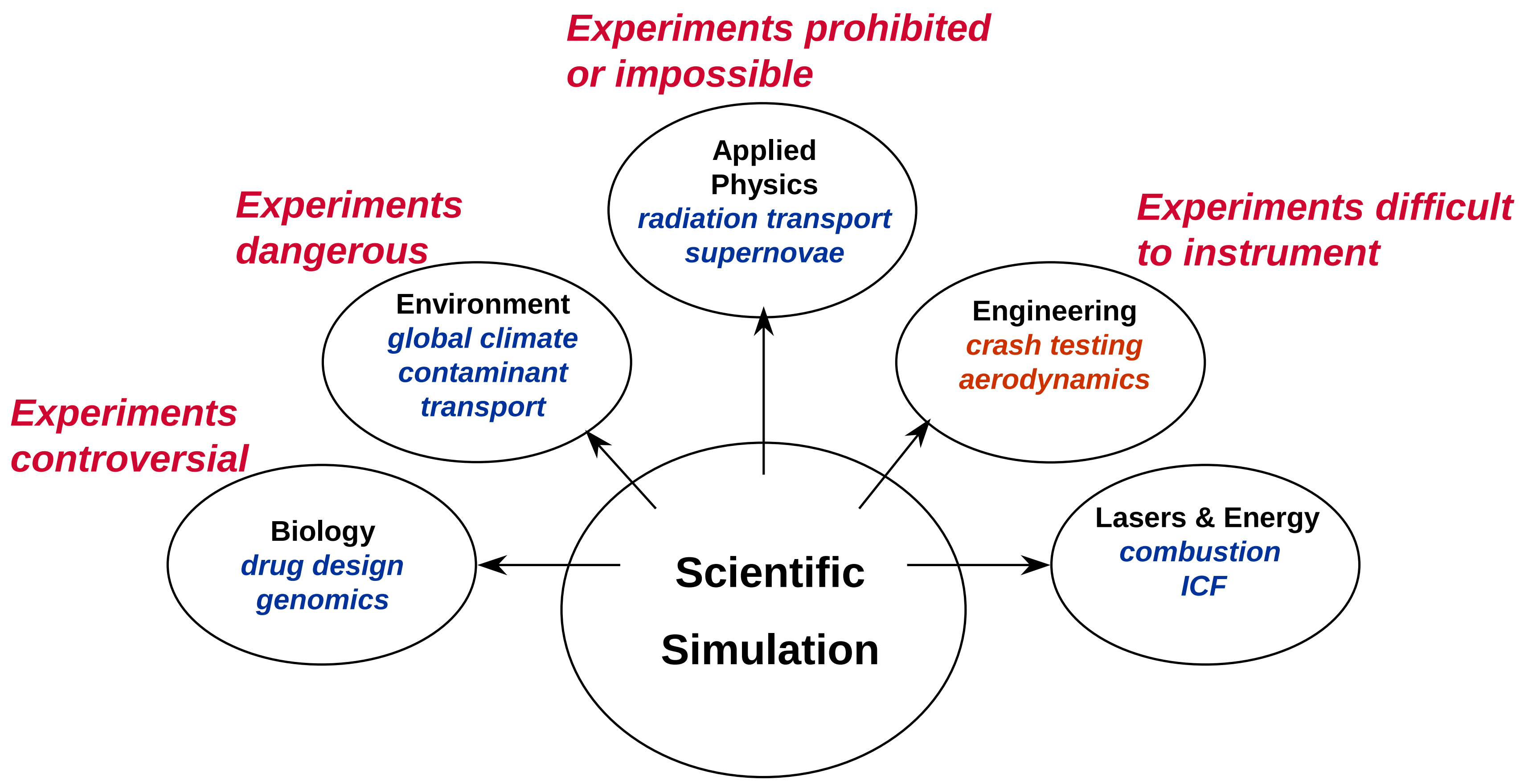
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been "sold"



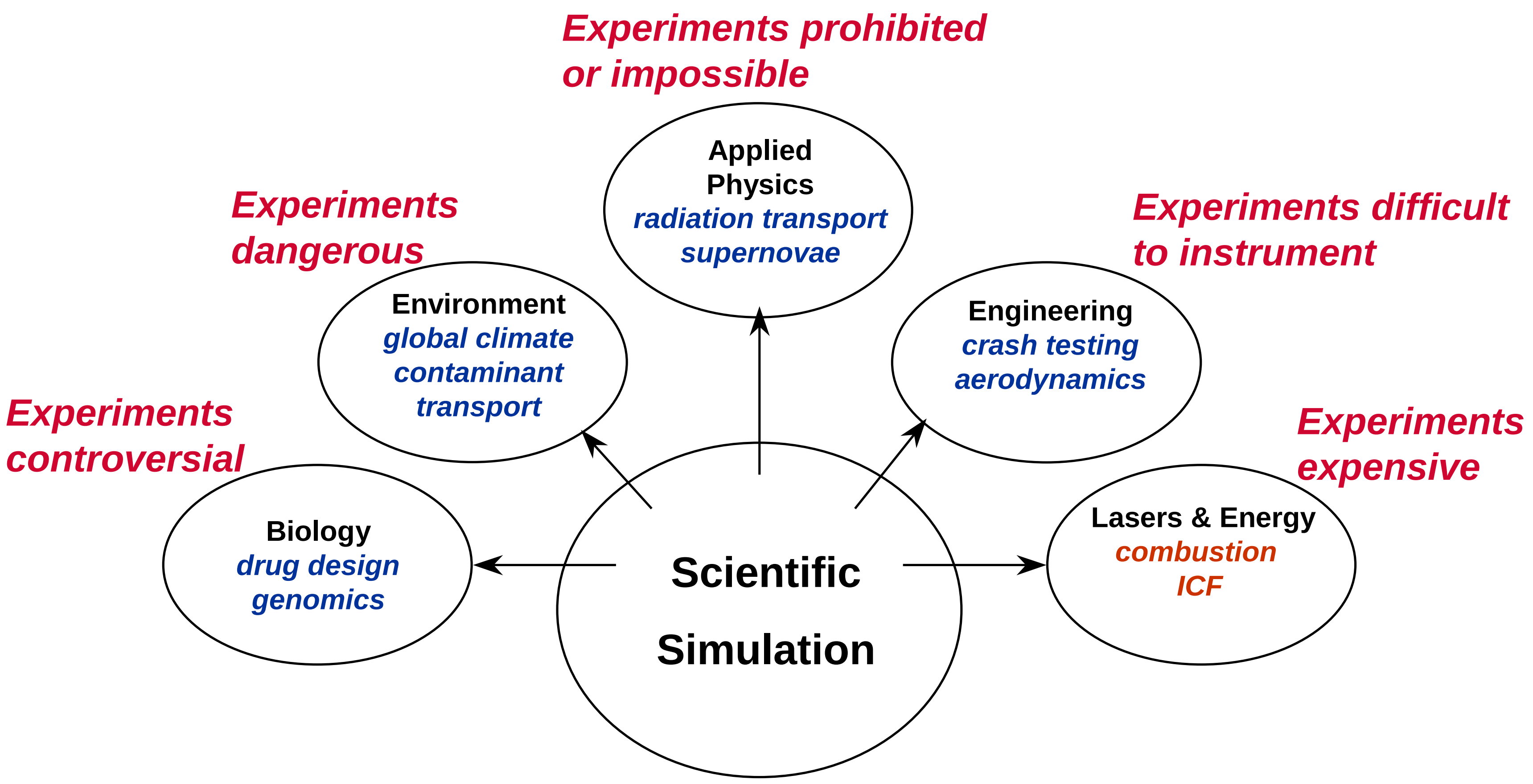
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been "sold"



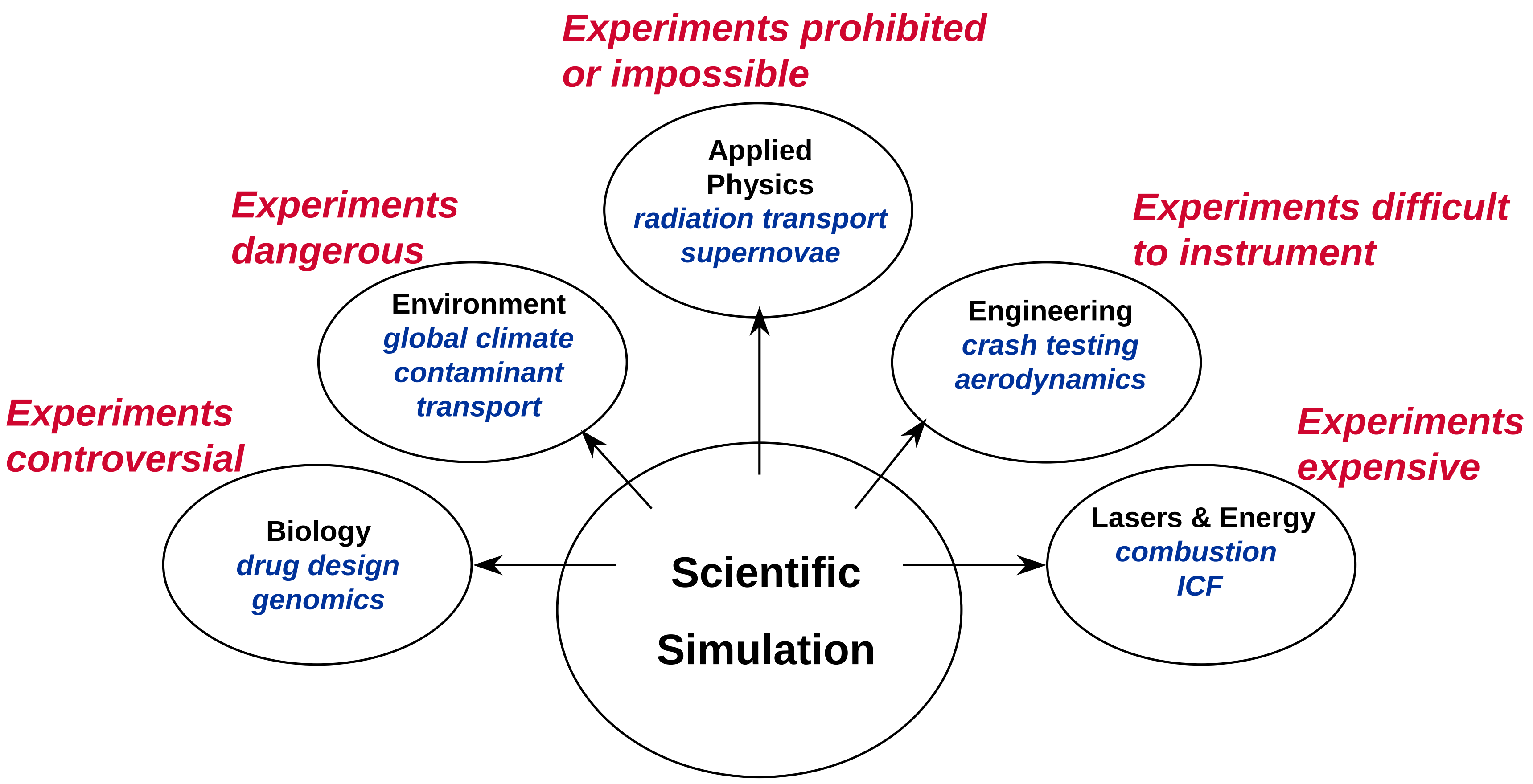
In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been "sold"



In these, and many other areas, simulation is an important complement to experiment.

Terascale simulation has been "sold"

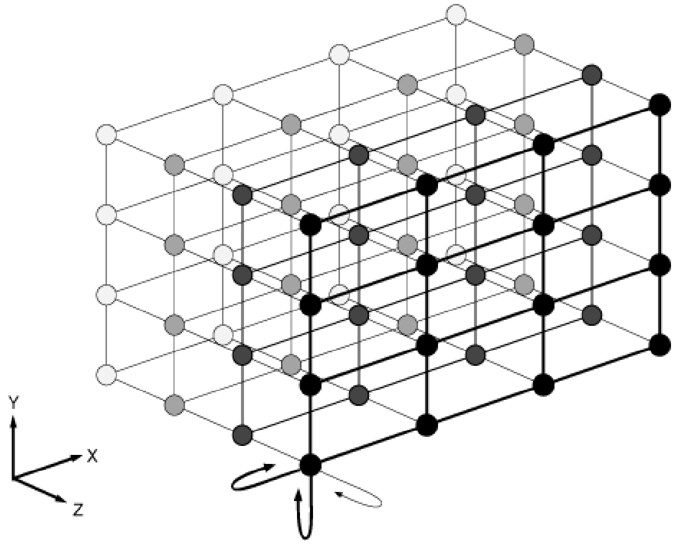


However, simulation is far from proven! To meet expectations, we need to handle problems of multiple physical scales.

Boundary conditions from architecture

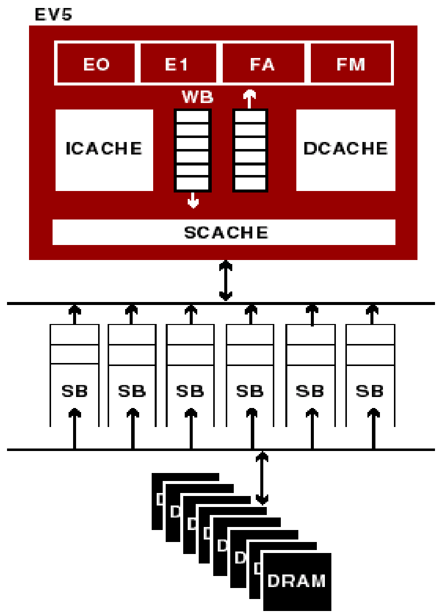
Algorithms must run on physically distributed memory units connected by message-passing network, each serving one or more processors with multiple levels of cache

"horizontal" aspects



network latency, BW, diameter

"vertical" aspects



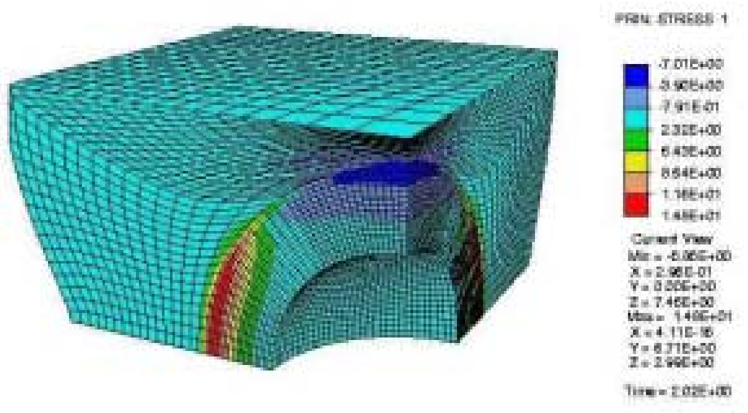
memory latency, BW; L/S (cache/reg) BW

Following the platforms …

- … Algorithms must be
  - highly concurrent and straightforward to load balance
  - not communication bound
  - cache friendly (temporal and spatial locality of reference)
  - highly scalable (in the sense of convergence)
- Goal for algorithmic scalability: fill up memory of arbitrarily large machines *while preserving constant running times* with respect to proportionally smaller problem on one processor
- Domain-decomposed multilevel methods “natural” for all of these
- Domain decomposition also “natural” for software engineering

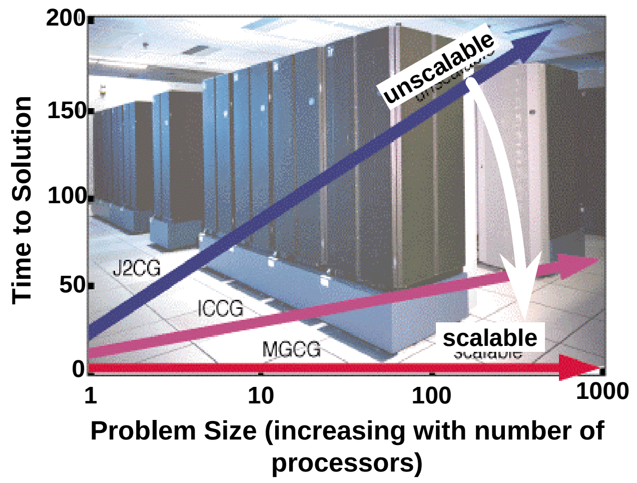
Keyword: "Optimal"

- Convergence rate nearly independent of discretization parameters
  - Multilevel schemes for rapid linear convergence of linear problems
  - Newton-like schemes for quadratic convergence of nonlinear problems
- Convergence rate as independent as possible of physical parameters
  - Continuation schemes
  - Physics-based preconditioning

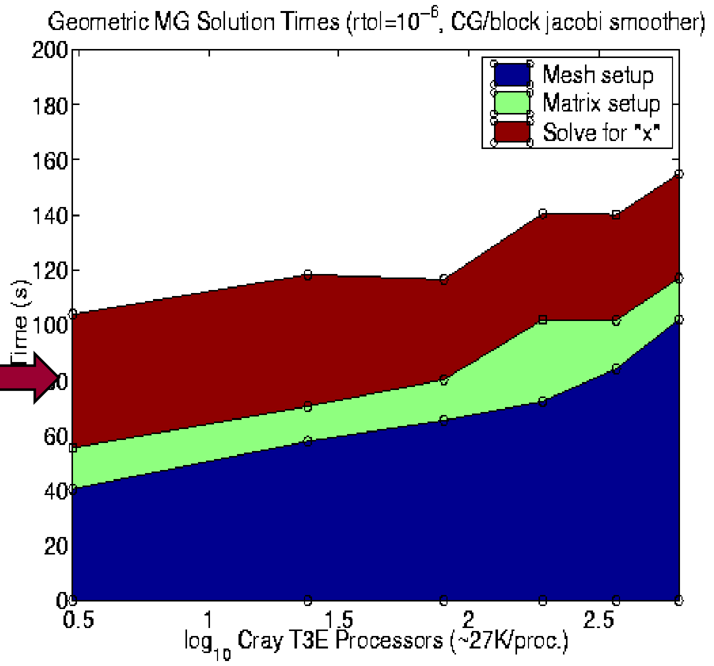


Steel/rubber composite

Parallel multigrid c/o M. Adams, Berkeley-Sandia



The solver is a key part, but not the only part, of the simulation that needs to be scalable

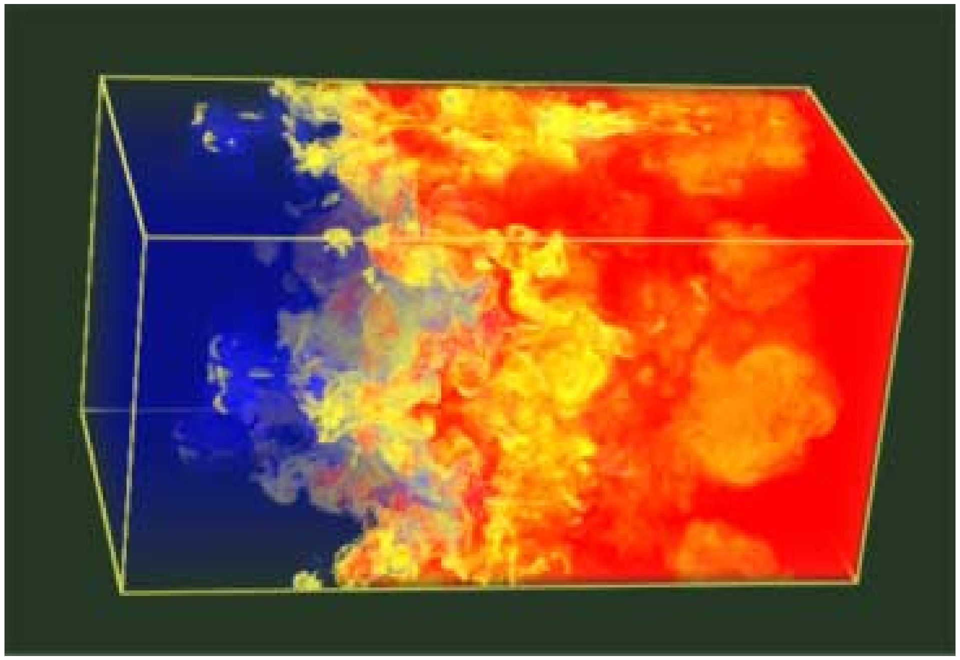


Why Optimal Algorithms?

- The more powerful the computer, the *greater* the premium on optimality
- Example:
  - Suppose *Alg1* solves a problem in time CN^2, where N is the input size
  - Suppose *Alg2* solves the same problem in time CN
  - Suppose that the machine on which *Alg1* and *Alg2* have been parallelized to run has 10,000 processors
- In constant time (compared to serial), *Alg1* can run a problem 100X larger, whereas *Alg2* can run a problem fully 10,000X larger
- Or, filling up the machine, *Alg1* takes 100X longer

Imperative: Multiple-scale Apps

- Multiple spatial scales
  - interfaces, fronts, layers
  - thin relative to domain size
- Multiple temporal scales
  - fast waves
  - small transit times relative to convection, diffusion, or group velocity dynamics



Richtmeyer-Meshkov instability, c/o A. Mirin, LLNL

- Analyst must isolate dynamics of interest and model the rest in a system that can be discretized over computable (modest) range of scales
- May lead to idealizations of local discontinuity or infinitely stiff subsystem requiring special treatment

Multiscale Stress on Algorithms

- **Spatial resolution stresses condition number**
  - **Ill-conditioning: small error in input may lead to large error in output**
  - **For self-adjoint linear systems cond no.** $\kappa = \| A \| \cdot \| A^{-1} \|$, **related to ratio of max to min eigenvalue**
  - **With improved resolution we approach the continuum limit of an unbounded inverse**
  - **For discrete Laplacian,** $\kappa = O(h^{-2})$
- **Standard iterative methods fail due to growth in iterations like** $O(\kappa)$ **or** $O(\sqrt{\kappa})$
- **Direct methods fail due to memory growth and bounded concurrency**
- **Solution is** ***hierarchical (multilevel) iterative methods***

Multiscale Stress on Algorithms, cont.

- Temporal resolution stresses stiffness
  - Stiffness: failure to track fastest mode may lead to exponentially growing error in other modes, related to ratio of max to min eigenvalue of *A*, in $\dot{y} = Ay$
  - By definition, multiple timescale problems contain phenomena of very different relaxation rates
  - Certain idealized systems (e.g., incomp NS) are infinitely stiff
- Number of steps to finite simulated time grows, to preserve stability, regardless of accuracy requirements
- Solution is to *step over* fast modes by assuming quasi-equilibrium
- Throws temporally stiff problems into spatially ill-conditioned regime

Multiscale Stress on Architecture

- Spatial resolution stresses memory size
  - *number* of floating point words
  - *precision* of floating point words
- Temporal resolution stresses clock rates
- Both stress interprocessor latency, and *together* they *severely* stress memory bandwidth
- *Less* severely stressed for PDEs, in principle, are memory latency and interprocessor bandwidth
  - Subject of *Europar2000* plenary (talk and paper available from my home page; URL later)
- Brute force not an option

Decomposition strategies for $\mathcal{L}u=f$ in Ω

- **Operator decomposition**

$$\mathcal{L} = \sum_k \mathcal{L}_k$$

- **Function space decomposition**

$$f = \sum_k f_k \Phi_k , u = \sum_k u_k \Phi_k$$

- **Domain decomposition**

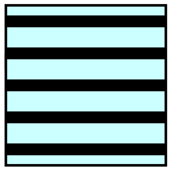
$$\Omega = \dagger_k \Omega_k$$

Consider, e.g., the implicitly discretized parabolic case

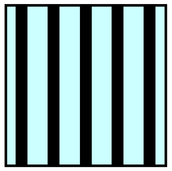
$$[\frac{I}{\tau} + \mathcal{L}_x + \mathcal{L}_y] u^{(k+1)} = \frac{I}{\tau} u^{(k)} + f$$

Operator decomposition

- Consider ADI



$$[\frac{I}{\tau/2} + L_x]u^{(k+1/2)} = [\frac{I}{\tau/2} - L_y]\boxed{u^{(k)}} + f$$



$$[\frac{I}{\tau/2} + L_y]\boxed{u^{(k+1)}} = [\frac{I}{\tau/2} - L_x]u^{(k+1/2)} + f$$

- Iteration matrix consists of four sequential (“multiplicative”) substeps per timestep
  - two sparse matrix-vector multiplies
  - two sets of unidirectional bandsolves
- Parallelism *within* each substep
- But global data exchanges *between* bandsolve substeps

Function space decomposition

- **Consider a spectral Galerkin method**

$$u(x,y,t) = \sum_{j=1}^{N} a_j(t)\Phi_j(x,y)$$

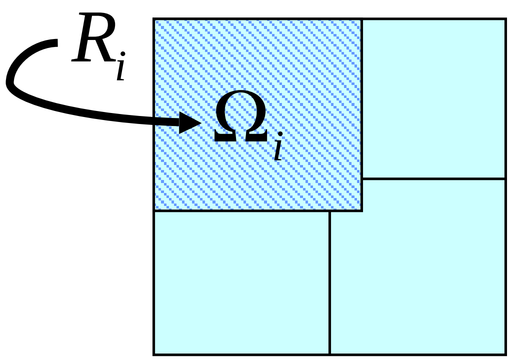
$$\tfrac{d}{dt}(\Phi_i, u) = (\Phi_i, \mathcal{L}u) + (\Phi_i, f), i = 1,\ldots,N$$

$$\textstyle\sum_j (\Phi_i, \Phi_j)\tfrac{da_j}{dt} = \sum_j (\Phi_i, \mathcal{L}\Phi_j)a_j + (\Phi_i, f), i = 1,\ldots,N$$

$$\tfrac{da}{dt} = M^{-1}Ka + M^{-1}f$$

- **System of ordinary differential equations**
- **Perhaps** $M \equiv [(\Phi_j, \Phi_i)], K \equiv [(\Phi_j, \mathcal{L}\Phi_i)]$ **are diagonal matrices**
- **Perfect parallelism across spectral index**
- **But global data exchanges to *transform back* to physical variables at each step**

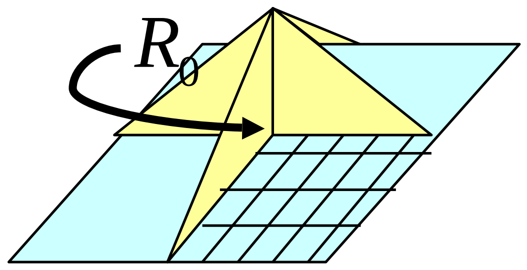
Domain decomposition



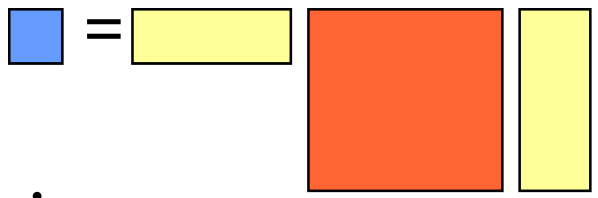
- Consider restriction and extension operators for subdomains, R_i, R_i^T, and for possible coarse grid, R_0, R_0^T
- Replace discretized $Au = f$ with

$$B^{-1}Au = B^{-1}f$$

$$B^{-1} = R_0^T A_0^{-1} R_0 + \sum_i R_i^T A_i^{-1} R_i$$



$$A_i = R_i A R_i^T$$



- Solve by a Krylov method, e.g., CG
- Matrix-vector multiplies with
  - parallelism on each subdomain
  - nearest-neighbor exchanges, global reductions
  - possible small global system (not needed for parabolic case)

Comparison

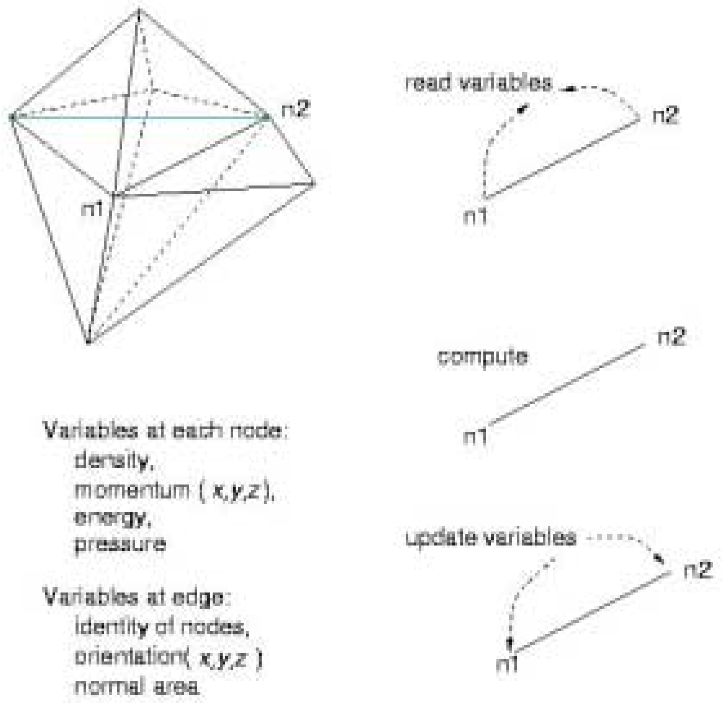
- Operator decomposition (ADI)
  - natural row-based assignment requires *all-to-all*, *bulk* data exchanges in each step (for transpose)
- Function space decomposition (Fourier)
  - natural mode-based assignment requires *all-to-all*, *bulk* data exchanges in each step (for transform)
- Domain decomposition (Schwarz)
  - natural domain-based assignment requires local (nearest neighbor) data exchanges, global reductions, and optional small global problem

Primary (DD) PDE solution kernels

- Vertex-based loops
  - state vector and auxiliary vector updates
- Edge-based “stencil op” loops
  - residual evaluation
  - approximate Jacobian evaluation
  - Jacobian-vector product (often replaced with matrix-free form, involving residual evaluation)
  - intergrid transfer (coarse/fine) in multilevel methods
- Subdomain-wise sparse, narrow-band recurrences
  - approximate factorization and back substitution
  - smoothing
- Vector inner products and norms
  - orthogonalization/conjugation
  - convergence progress and stability checks

Illustration of edge-based loop

- Vertex-centered grid
- Traverse by edges
  - load vertex values
  - compute intensively
    - e.g., for compressible flows, solve 5x5 eigen-problem for character-istic directions and speeds of each wave
  - store flux contributions at vertices
- Each vertex appears in approximately 15 flux computations (for tets)



Complexities of PDE kernels

- Vertex-based loops
  - work and data closely proportional
  - pointwise concurrency, no communication
- Edge-based “stencil op” loops
  - large ratio of work to data
  - colored edge concurrency; local communication
- Subdomain-wise sparse, narrow-band recurrences
  - work and data closely proportional
- Vector inner products and norms
  - work and data closely proportional
  - pointwise concurrency; global communication

Potential architectural stresspoints

- **Vertex-based loops:**
  - **memory bandwidth**
- **Edge-based "stencil op" loops:**
  - **load/store (register-cache) bandwidth**
  - **internode bandwidth**
- **Subdomain-wise sparse, narrow-band recurrences:**
  - **memory bandwidth**
- **Inner products and norms:**
  - **memory bandwidth**
  - **internode latency, network diameter**
- **ALL STEPS:**
  - **memory latency, unless good locality is consciously built-in**

Theoretical scaling of domain decomposition
(for three common network topologies*)

- **With logarithmic-time (hypercube- or tree-based) global reductions and scalable nearest neighbor interconnects:**
  - **optimal number of processors scales *linearly* with problem size (“*scalable*”, assumes one subdomain per processor)**
- **With power-law-time (3D torus-based) global reductions and scalable nearest neighbor interconnects:**
  - **optimal number of processors scales as *three-fourths* power of problem size (“*almost scalable*”)**
- **With linear-time (common bus) network:**
  - **optimal number of processors scales as *one-fourth* power of problem size (**not* scalable*)**
  - **bad news for conventional Beowulf clusters, but see 2000 & 2001 Bell Prize “price-performance awards” using multiple commodity NICs per Beowulf node!**

* subject of DD’98 proceedings paper (on-line)

Basic Concepts

- Iterative correction (including CG and MG)
- Schwarz preconditioning

“Advanced” Concepts

- Newton-Krylov-Schwarz
- Nonlinear Schwarz

Iterative correction

- **The most basic idea in iterative methods**

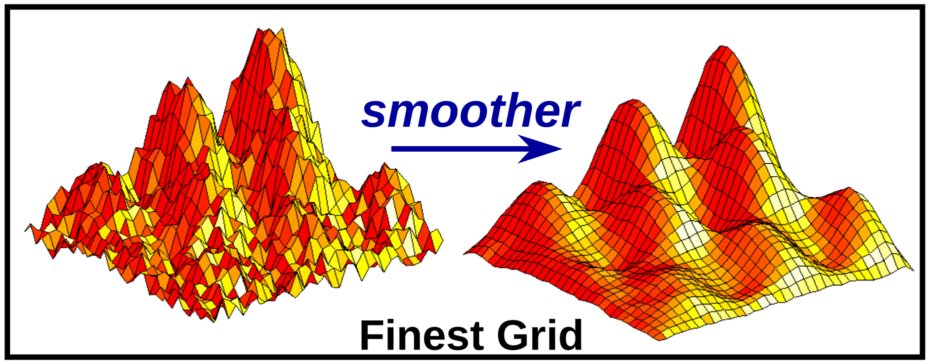
$$u \leftarrow u + B^{-1}(f - Au)$$

- **Evaluate residual accurately, but solve approximately, where B^{-1} is an approximate inverse to A**
- **A sequence of complementary solves can be used, e.g., with B_1 first and then B_2 one has**

$$u \leftarrow u + [B_1^{-1} + B_2^{-1} - B_2^{-1}AB_1^{-1}](f - Au)$$

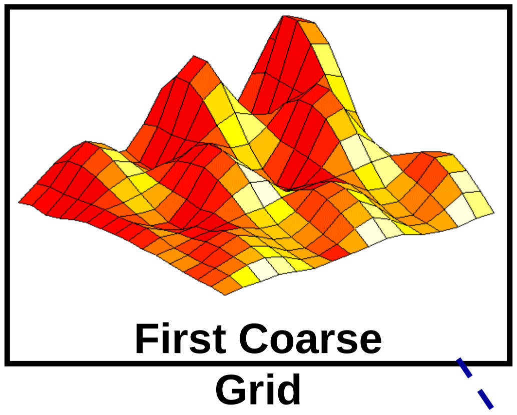
- **Optimal polynomials of $(B^{-1}A)$ lead to various *preconditioned Krylov methods***
- **Scale recurrence, e.g., with $B_2^{-1} = R^T(RAR^T)^{-1}R$, leads to *multilevel methods***

Multilevel Preconditioning



A Multigrid V-cycle

Restriction
transfer from fine to coarse grid



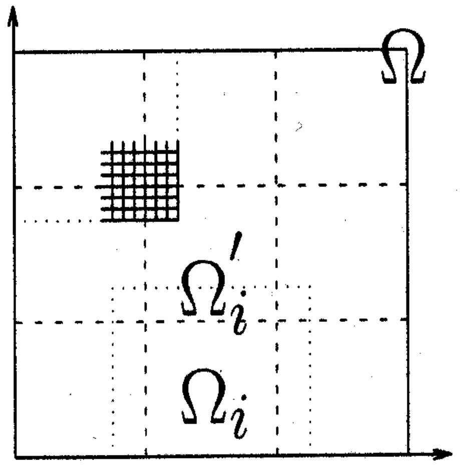
coarser grid has fewer cells (less work & storage)

Prolongation
transfer from coarse to fine grid

Recursively apply this idea until we have an easy problem to solve

Schwarz Preconditioning

- **Given $A x = b$, partition x into subvectors, corresp. to subdomains Ω_i of the domain Ω of the PDE, nonempty, possibly overlapping, whose union is all of the elements of $x \in \Re^n$**

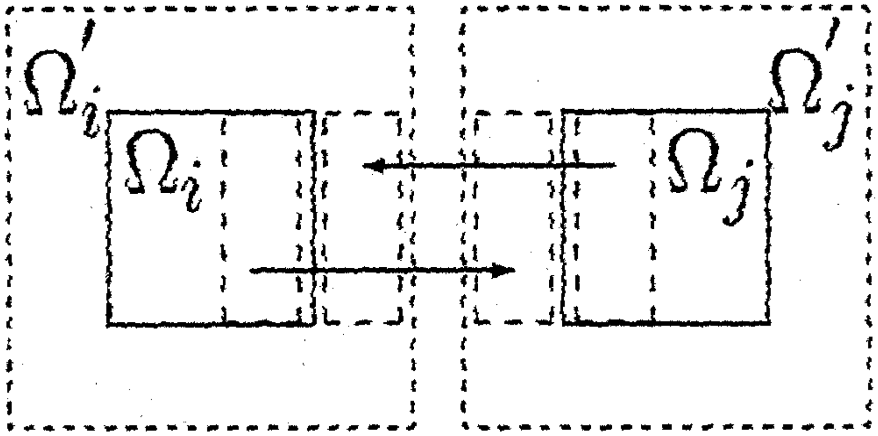


- **Let Boolean rectangular matrix R_i extract the i^{th} subset of x :**

$$x_i = R_i x$$

- **Let $A_i = R_i A R_i^T$**

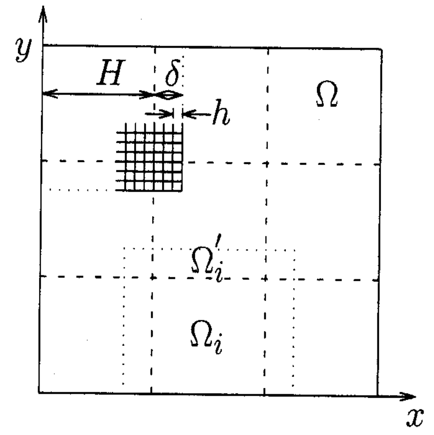
$$B^{-1} = \sum_i R_i^T A_i^{-1} R_i$$



The Boolean matrices are gather/scatter operators, mapping between a global vector and its subdomain support

Iteration count estimates from the Schwarz theory

- Krylov-Schwarz iterative methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system
- In terms of N and P, where for d-dimensional isotropic problems, $N=h^{-d}$ and $P=H^{-d}$, for mesh parameter h and subdomain diameter H, iteration counts may be estimated as follows:



Preconditioning Type	in 2D	in 3D
Point Jacobi	$O(N^{1/2})$	$O(N^{1/3})$
Domain Jacobi (δ=0)	$O((NP)^{1/4})$	$O((NP)^{1/6})$
1-level Additive Schwarz	$O(P^{1/2})$	$O(P^{1/3})$
2-level Additive Schwarz	$O(1)$	$O(1)$

Newton-Krylov-Schwarz

Popularized in parallel Jacobian-free form under this name by Cai, Gropp, Keyes & Tidriri (1994)

Newton
nonlinear solver
asymptotically quadratic

Krylov
accelerator
spectrally adaptive

Schwarz
preconditioner
parallelizable

Jacobian-Free Newton-Krylov Method

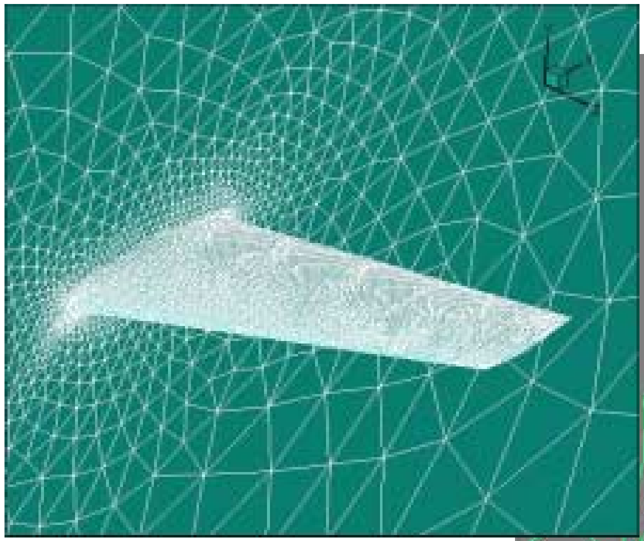
- **In the Jacobian-Free Newton-Krylov (JFNK) method, a Krylov method solves the linear Newton correction equation, requiring Jacobian-vector products**
- **These are approximated by the Fréchet derivatives**

$$J(u)v \approx \frac{1}{\varepsilon}[F(u+\varepsilon v)-F(u)]$$

so that the actual Jacobian elements are never explicitly needed, where ε is chosen with a fine balance between approximation and floating point rounding error

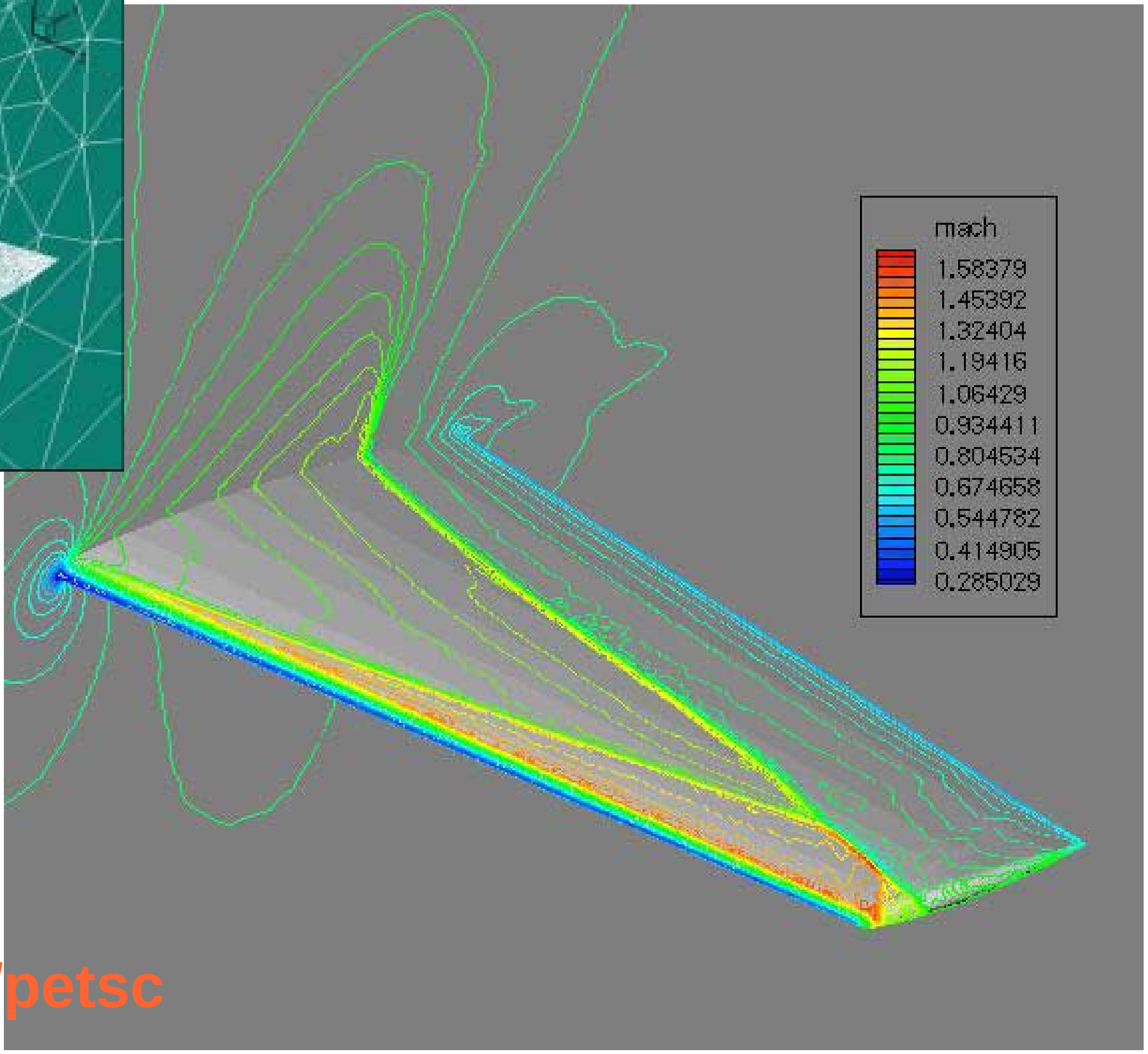
- **Schwarz preconditions, using approx. Jacobian**

Computational Aerodynamics



mesh c/o D. Mavriplis, ICASE

Transonic "Lambda" Shock, Mach contours on surfaces

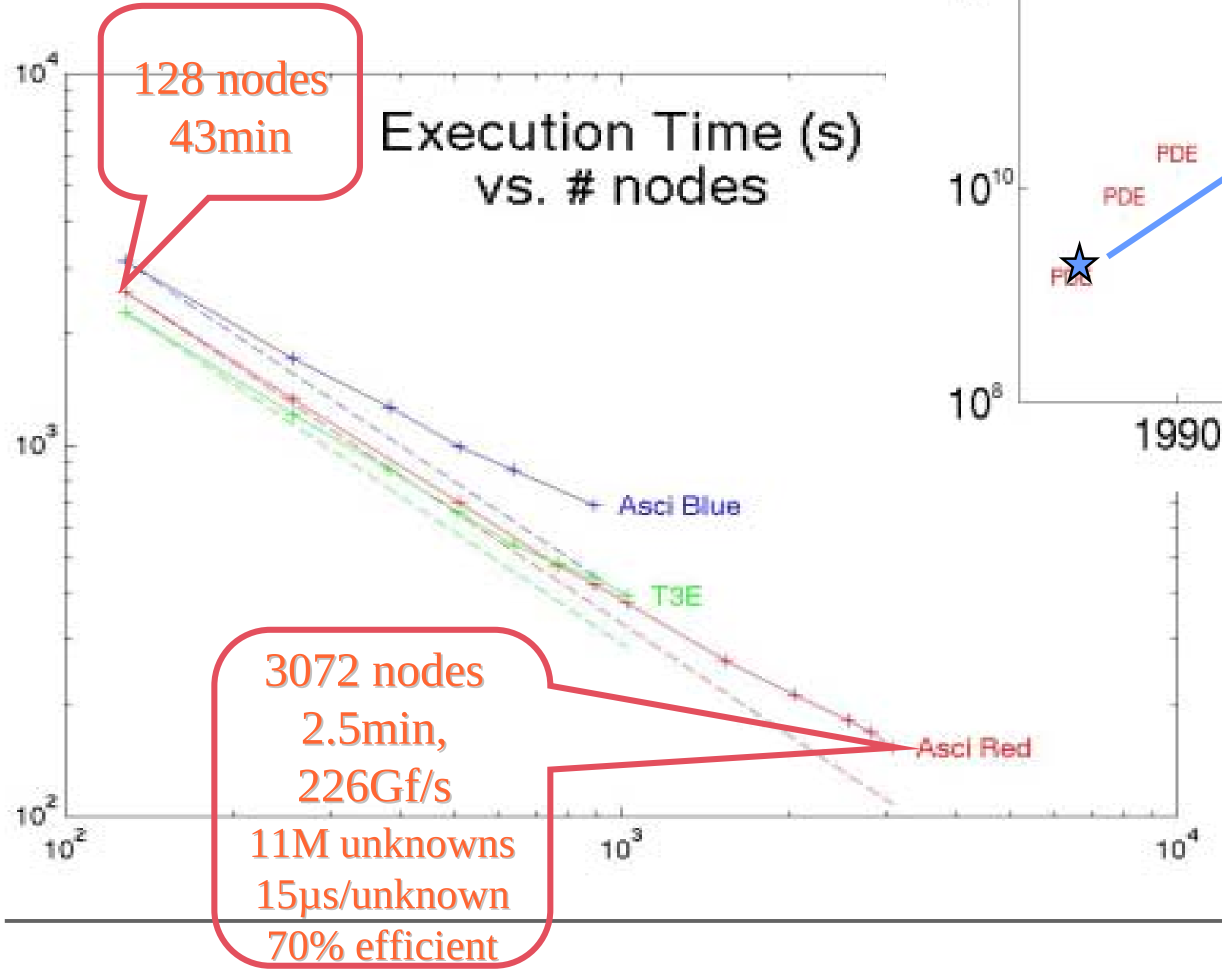


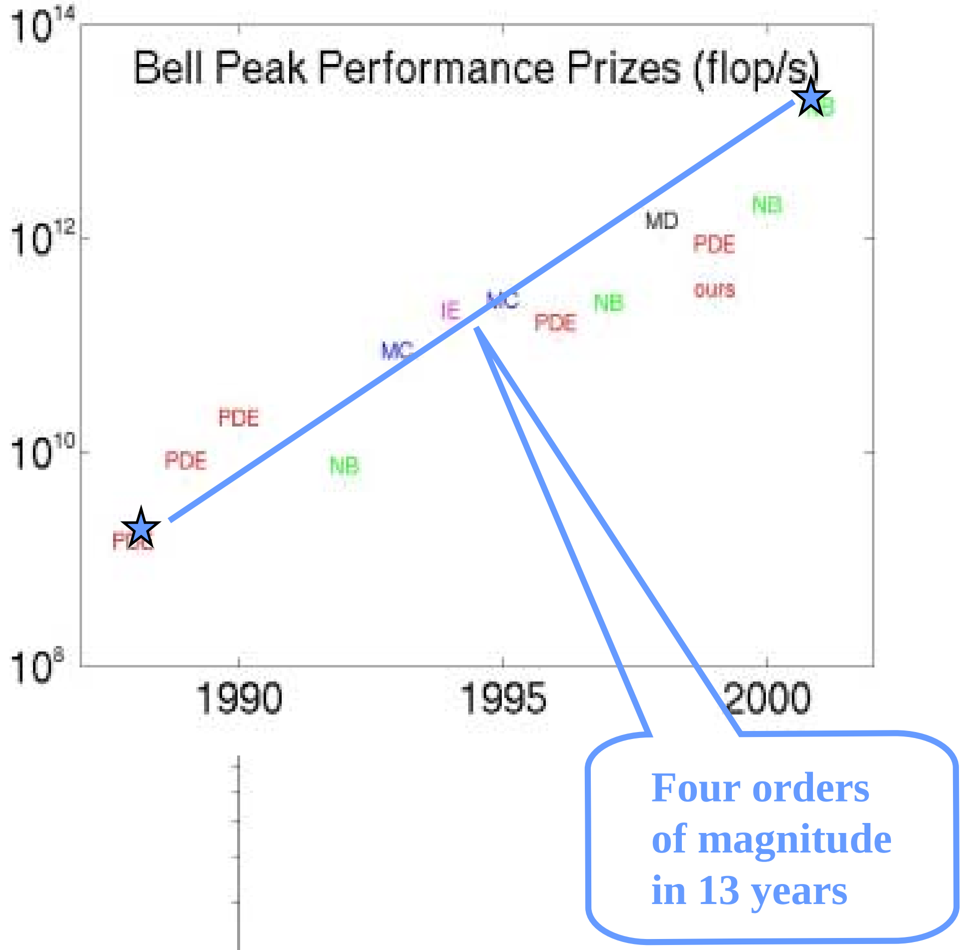
Implemented in PETSc

www.mcs.anl.gov/petsc

Fixed-size Parallel Scaling Results

This scaling study, featuring our widest range of processor number, was done for the *incompressible* case.

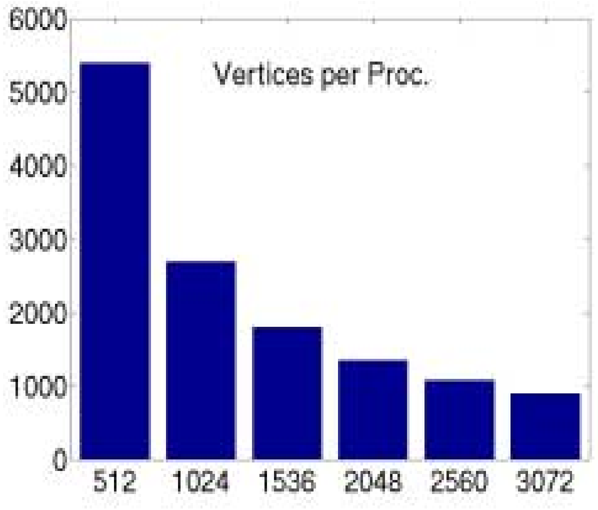


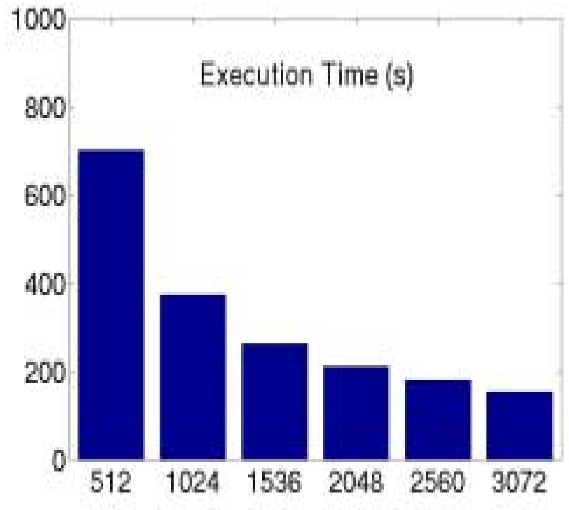


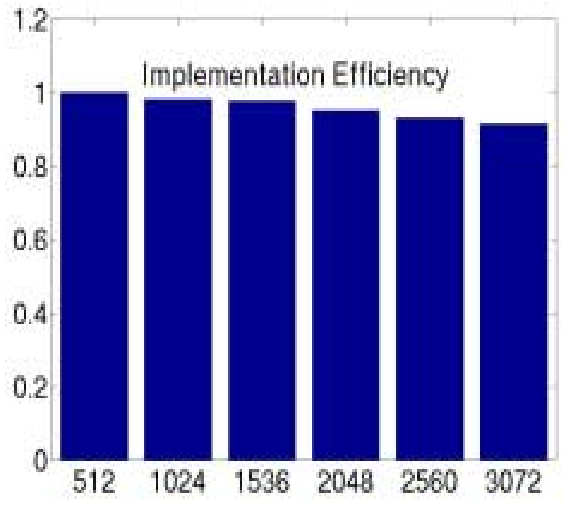
c/o K. Anderson, W. Gropp, D. Kaushik, D. Keyes and B. Smith

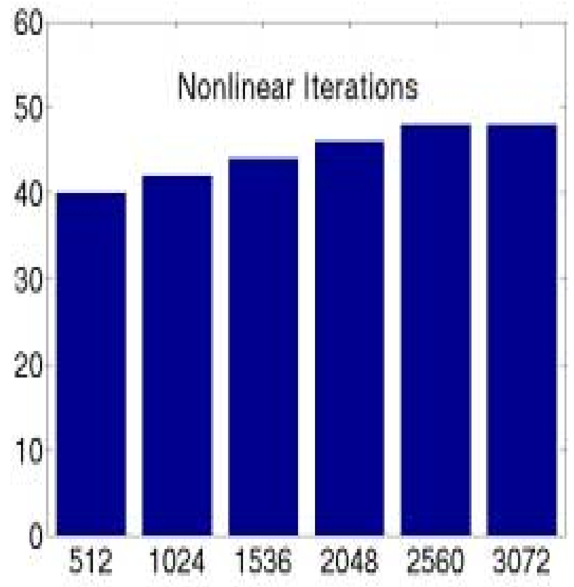
Fixed-size Parallel Scaling Results on ASCI Red

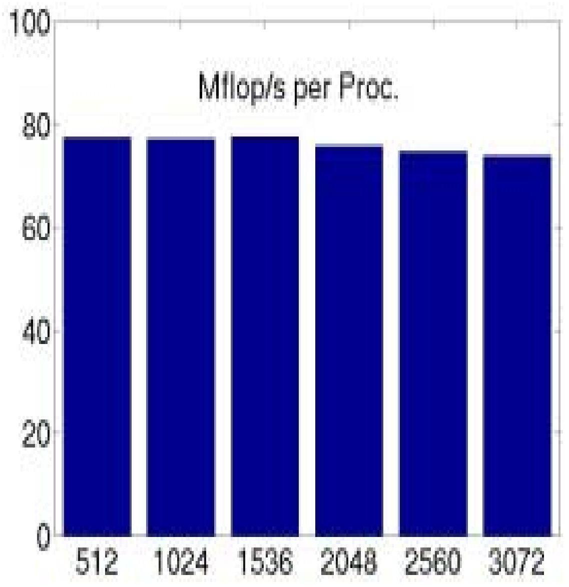
ONERA M6 Wing Test Case, Tetrahedral grid of 2.8 million vertices on up to 3072 ASCI Red Nodes (Pentium Pro 333 MHz processors)

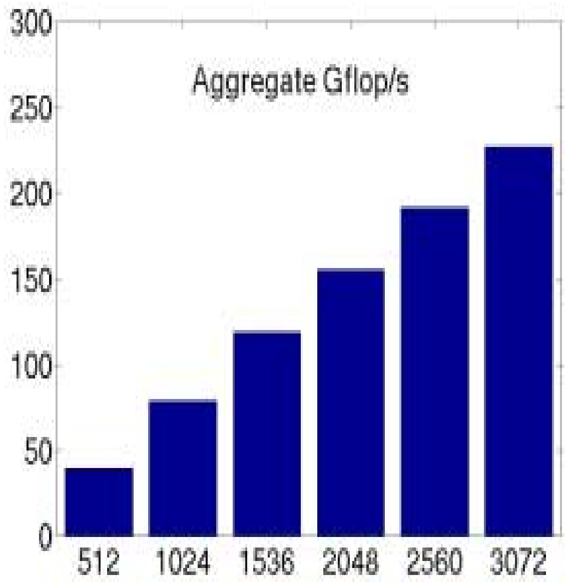






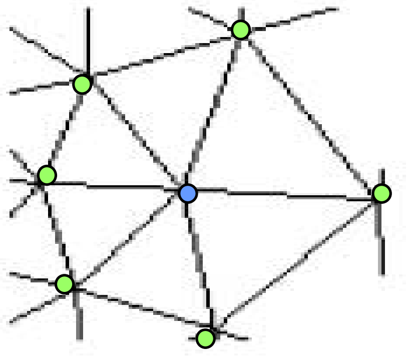


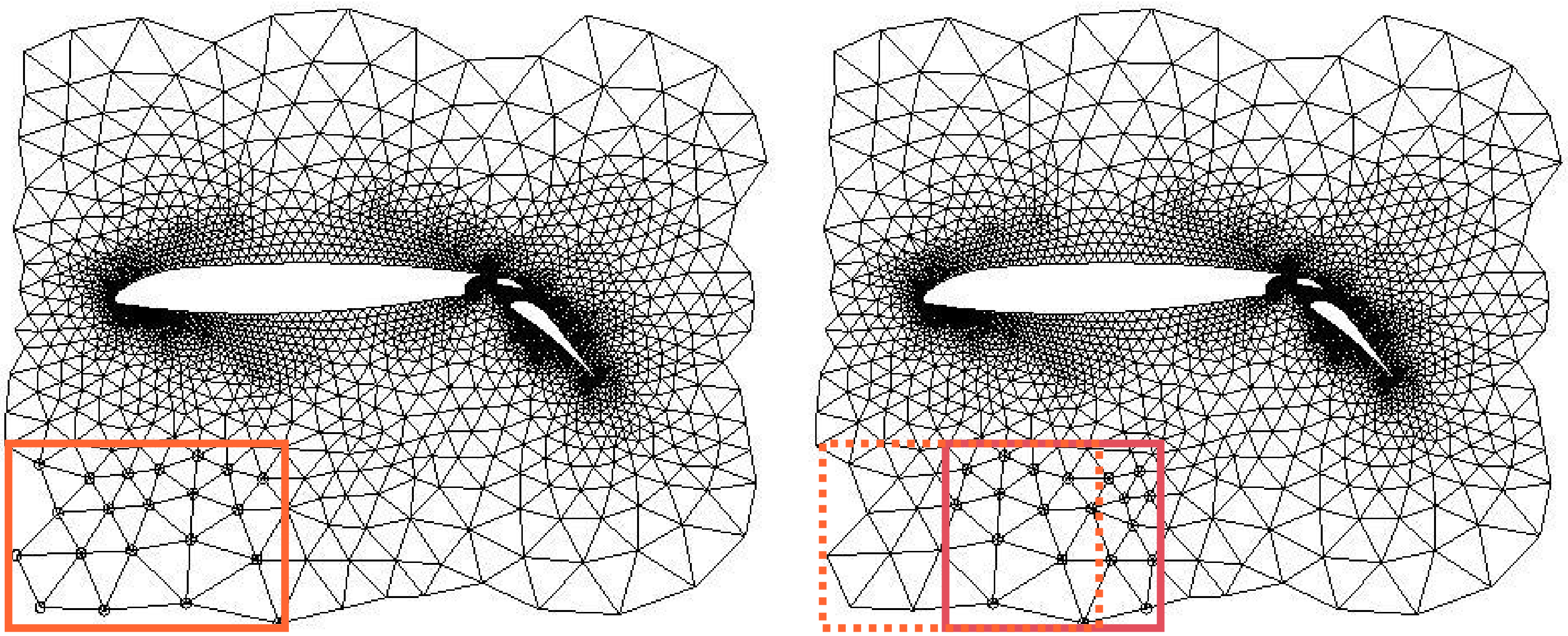




PDE Workingsets

- **Smallest: data for single stencil**
- **Largest: data for entire subdomain**
- **Intermediate: data for a neighborhood collection of stencils, reused as possible**





Improvements Resulting from Locality Reordering

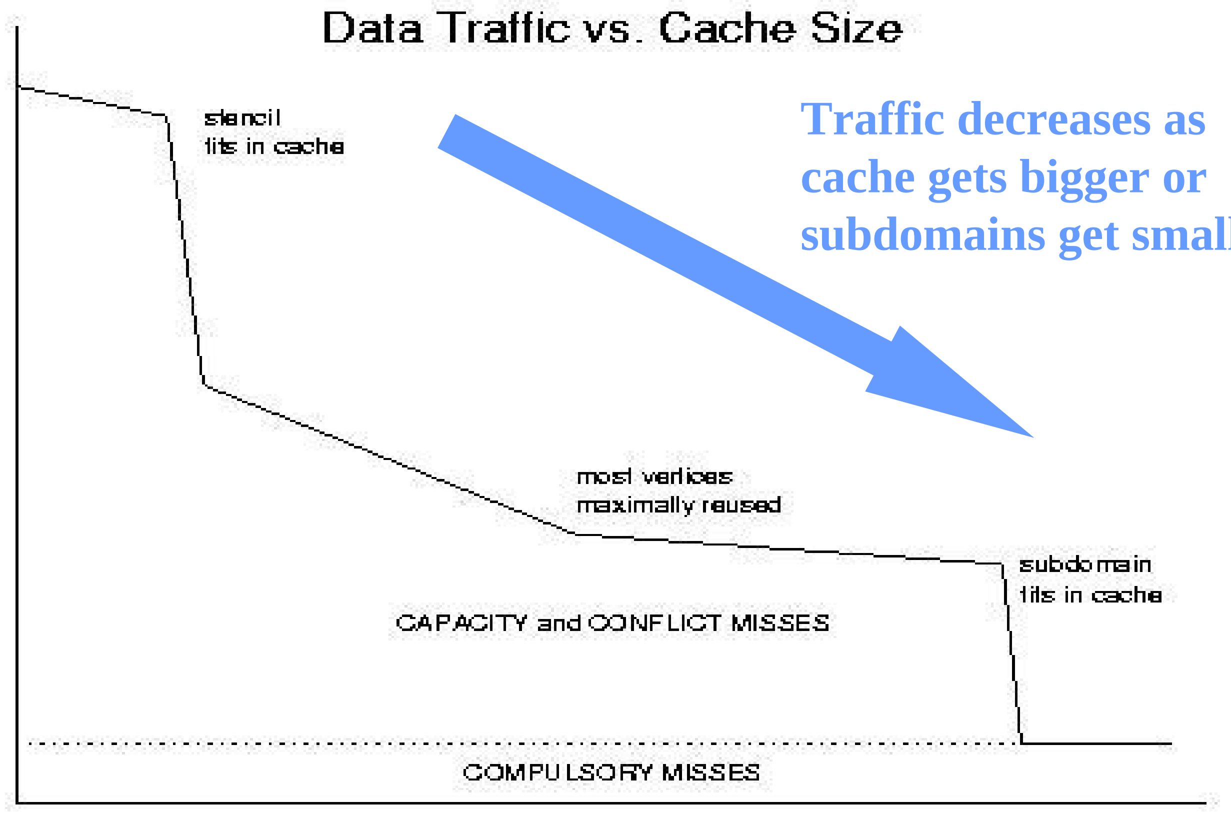
Factor of Five!

Processor	Clock MHz	Peak Mflop/s	Opt. % of Peak	Opt. Mflop/s	Reord. Only Mflop/s	Interl. only Mflop/s	Orig. Mflop/s	Orig. % of Peak
R10000	250	500	25.4	127	74	59	26	5.2
P3	200	800	20.3	163	87	68	32	4.0
P2SC (2 card)	120	480	21.4	101	51	35	13	2.7
P2SC (4 card)	120	480	24.3	117	59	40	15	3.1
604e	332	664	9.9	66	43	31	[illegible]	2.3
Alpha 21164	450	900	[illegible]	[illegible]	39	[illegible]	[illegible]	1.6
Alpha [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	47	[illegible]	[illegible]	1.3
Ultra II	[illegible]	[illegible]	[illegible]	[illegible]	42	[illegible]	[illegible]	3.0
Ultra [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	25	3.5
Ultra [illegible]	[illegible]	[illegible]	[illegible]	71	47	36	20	2.5
Pent. [illegible]	400	400	20.8	83	52	47	33	8.3
Pent. II/NT	400	400	19.5	78	49	49	31	7.8
Pent. Pro	200	200	21.0	42	27	26	16	8.0
Pent. Pro	333	333	18.8	60	40	36	21	6.3

See poster by D. Kaushik

Cache Traffic for PDEs

- **As successive workingsets "drop" into a level of memory, capacity (and with effort conflict) misses disappear, leaving only compulsory, reducing demand on main memory bandwidth**



Traffic decreases as cache gets bigger or subdomains get smaller

Nonlinear Schwarz preconditioning

- **Nonlinear Schwarz has Newton both *inside* and *outside* and is fundamentally Jacobian-free**
- **It replaces** $F(u)=0$ **with a new nonlinear system possessing the same root,** $\Phi(u)=0$
- **Define a correction** $\delta_i(u)$ **to the** i^{th} **partition (e.g., subdomain) of the solution vector by solving the following local nonlinear system:**

$$R_i F(u+\delta_i(u))=0$$

where $\delta_i(u)\in\Re^n$ **is nonzero only in the components of the** i^{th} **partition**

- **Then sum the corrections:** $\Phi(u)\equiv\sum_i \delta_i(u)$

Nonlinear Schwarz, cont.

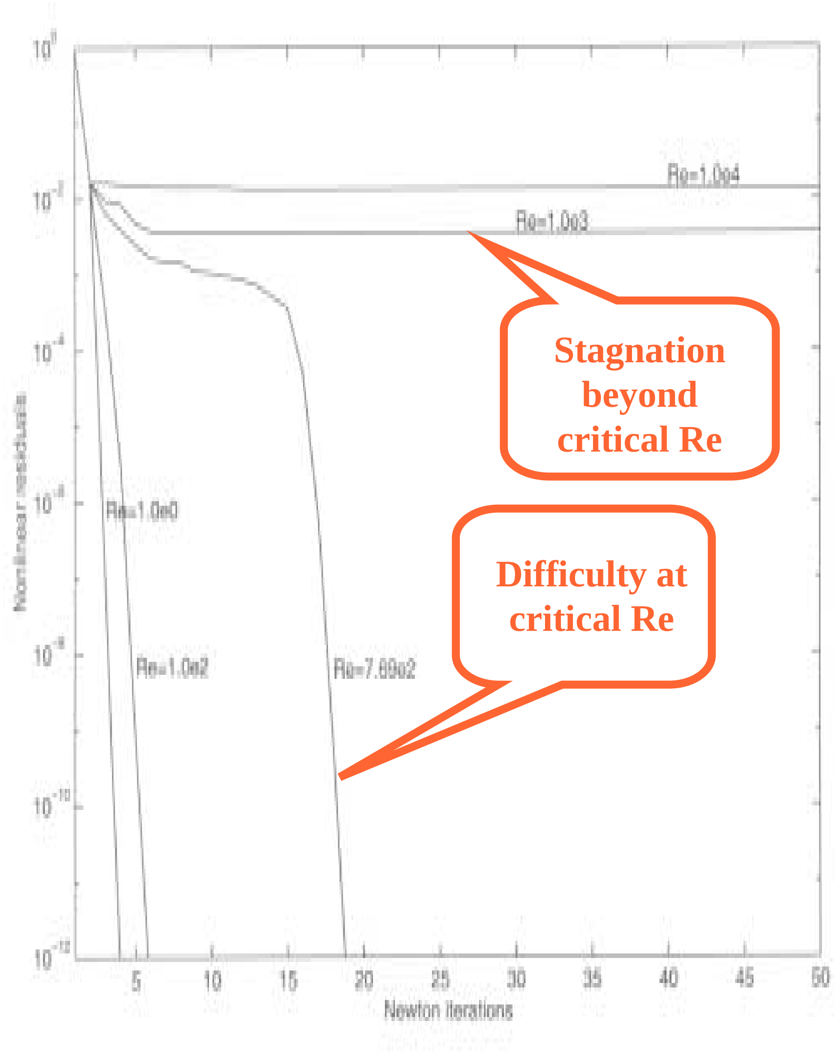
- **It is simple to prove that if the Jacobian of $F(u)$ is nonsingular in a neighborhood of the desired root then $\Phi(u)=0$ and $F(u)=0$ have the same unique root**
- **To lead to a Jacobian-free Newton-Krylov algorithm we need to be able to evaluate for any $u,v\in\Re^n$:**
  - **The residual $\Phi(u)=\sum_i \delta_i(u)$**
  - **The Jacobian-vector product $\Phi(u)'v$**
- **Remarkably, (Cai-Keyes, 2000) it can be shown that**

$$\Phi'(u)v \approx \sum_i (R_i^T J_i^{-1} R_i) Jv$$

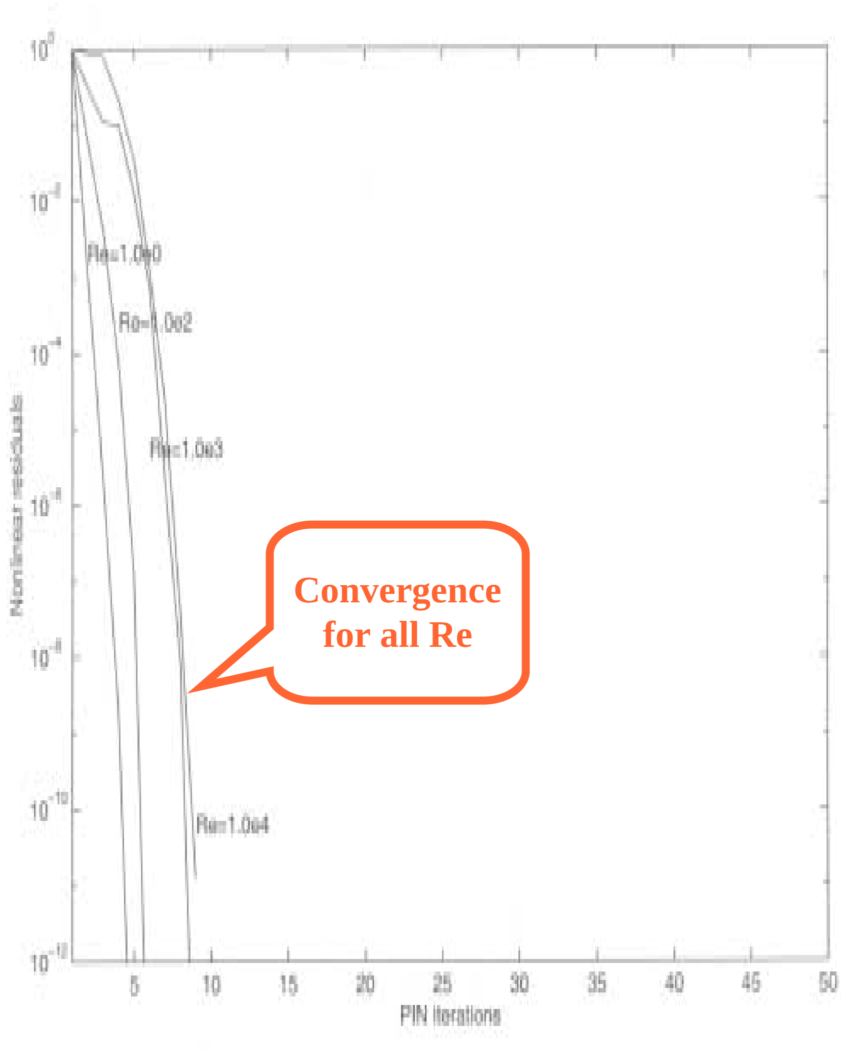
where $J=F'(u)$ and $J_i = R_i J R_i^T$

- **All required actions are available in terms of $F(u)$!**

Experimental example of nonlinear Schwarz



Newton’s method



Additive Schwarz Preconditioned Inexact Newton (ASPIN)

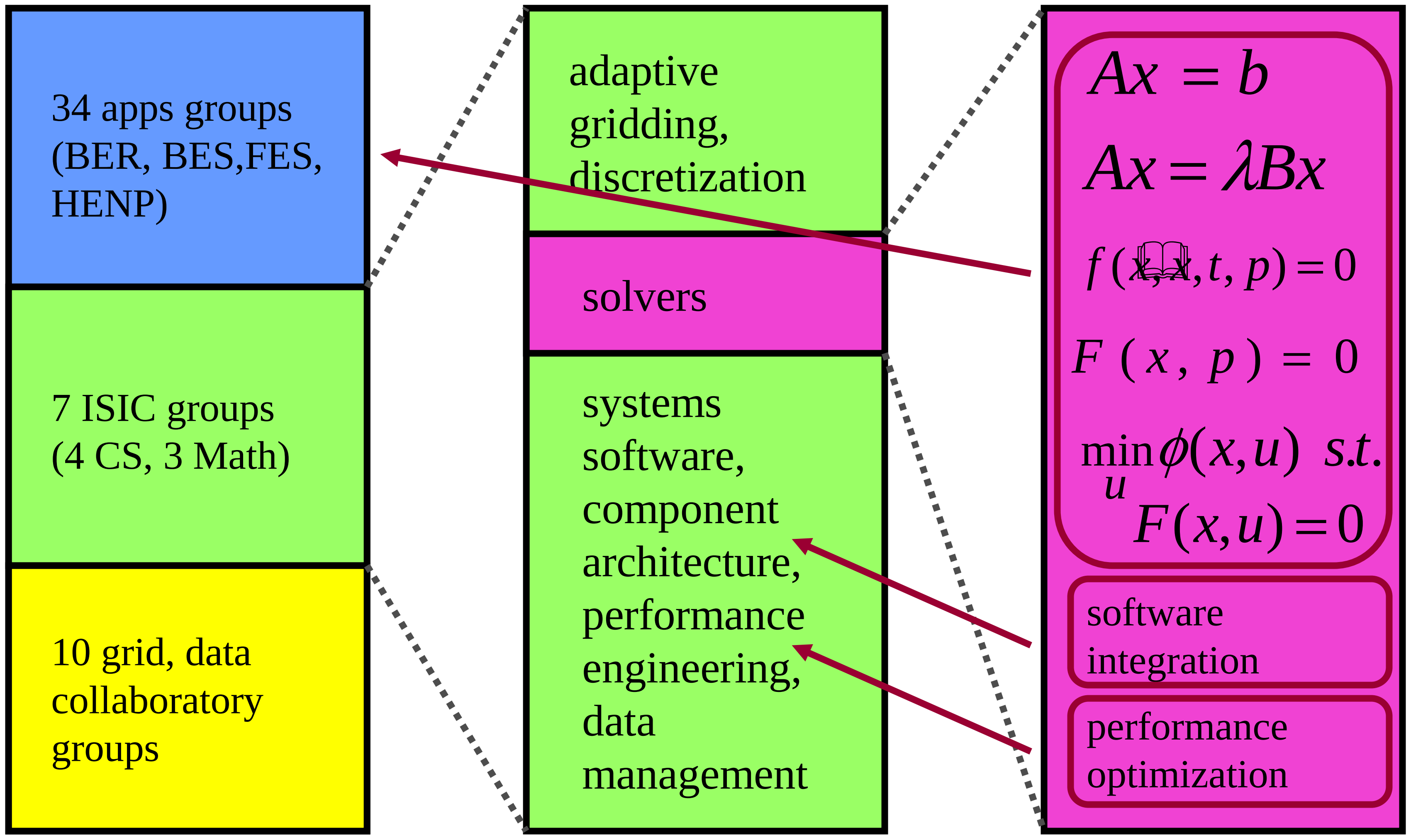
SciDAC

Scientific Discovery through Advanced Computing

- **Lab-university collaborations to develop reusable software “solutions” and partner with application groups**
- **For FY2002, 51 new projects at $57M/year total**
  - **Approximately one-third for applications**
  - **A third for integrated software infrastructure centers**
  - **A third for grid infrastructure and collaboratories**
- **5 Tflop/s IBM SP platforms “Seaborg” at NERSC (#3 in latest “Top 500”) and “Cheetah” at ORNL (being installed now) available for SciDAC**

SciDAC

Scientific Discovery through Advanced Computing


34 apps groups (BER, BES,FES, HENP)
7 ISIC groups (4 CS, 3 Math)
10 grid, data collaboratory groups
adaptive gridding, discretization
solvers
systems software, component architecture, performance engineering, data management
$Ax = b$
$Ax = \lambda Bx$
$f(\dot{x}, x, t, p) = 0$
$F(x, p) = 0$
$\min_u \phi(x,u)$ s.t. $F(x,u) = 0$
software integration
performance optimization

Other SciDAC Salishan'02 attendees

- **HQ**
  - David Bader (BER), Fred Johnson (MICS)
- **Apps**
  - Buddy Bland (HENP), Bob Harrison (BES), Chris Johnson (FES)
- **PERC ISIC**
  - David Bailey, Jeff Hollingsworth, Allen Maloney, Dan Reed, Allen Snavely, Jeff Vetter, Pat Worley
- **TSTT ISIC**
  - David Brown, Lori Freitag
- **CCA ISIC**
  - Lois McInnes
- **Scalable Software ISIC**
  - Al Geist
- **TOPS ISIC**
  - Jack Dongarra, David Keyes

Introducing "Terascale Optimal PDE Simulations" (TOPS) ISIC

Nine institutions, $18M, five years, 24 co-PIs

TOPS

Terascale Optimal PDE Simulations

TOPS

- Not just algorithms, but vertically integrated software suites
- Portable, scalable, extensible, tunable, modular implementations
- Starring PETSc and hypre, among other existing packages
- Driven by three applications SciDAC groups
  - LBNL-led “21st Century Accelerator” designs
  - ORNL-led core collapse supernovae simulations
  - PPPL-led magnetic fusion energy simulations

intended for many others

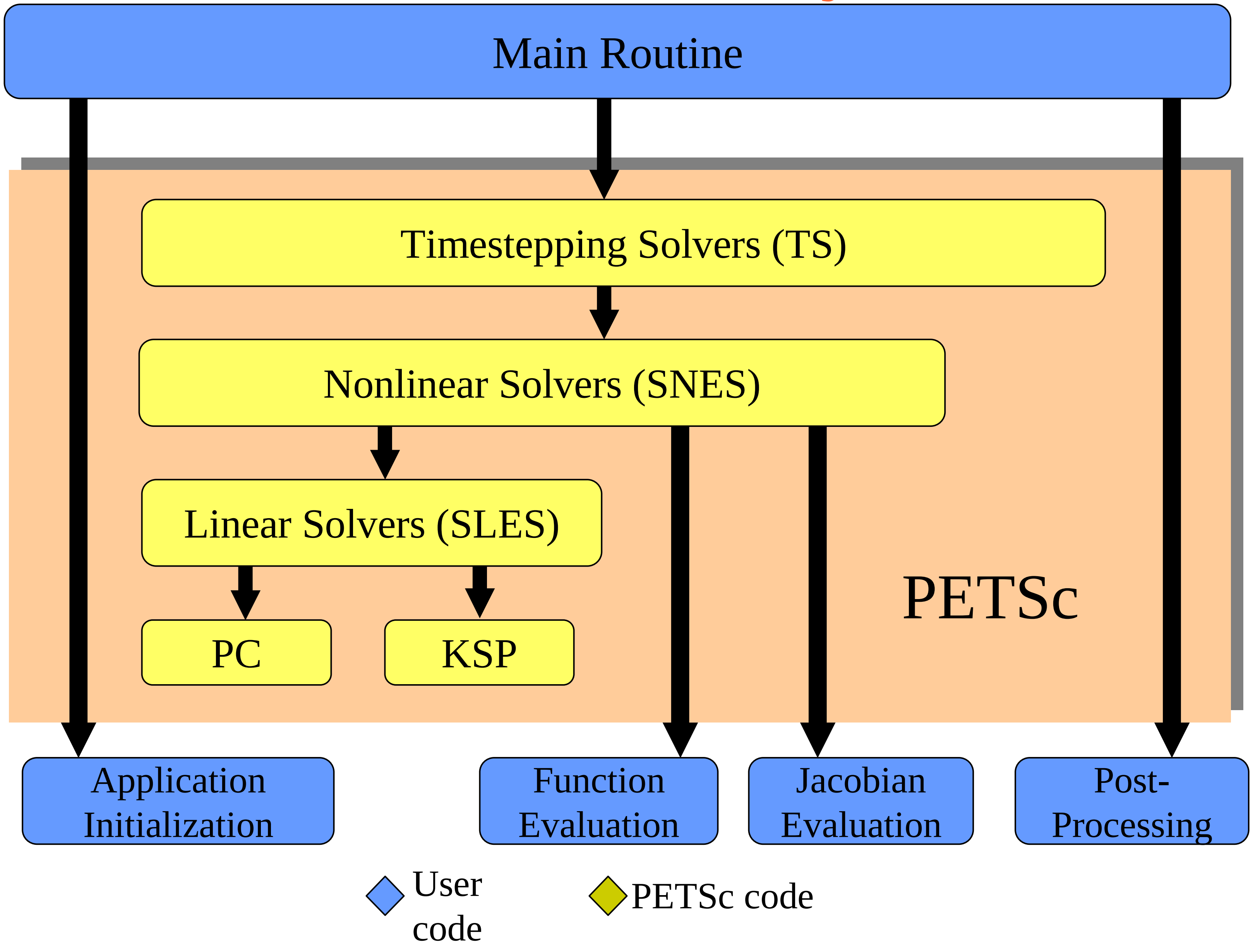
Background of PETSc Library

(in which FUN3D example was implemented)

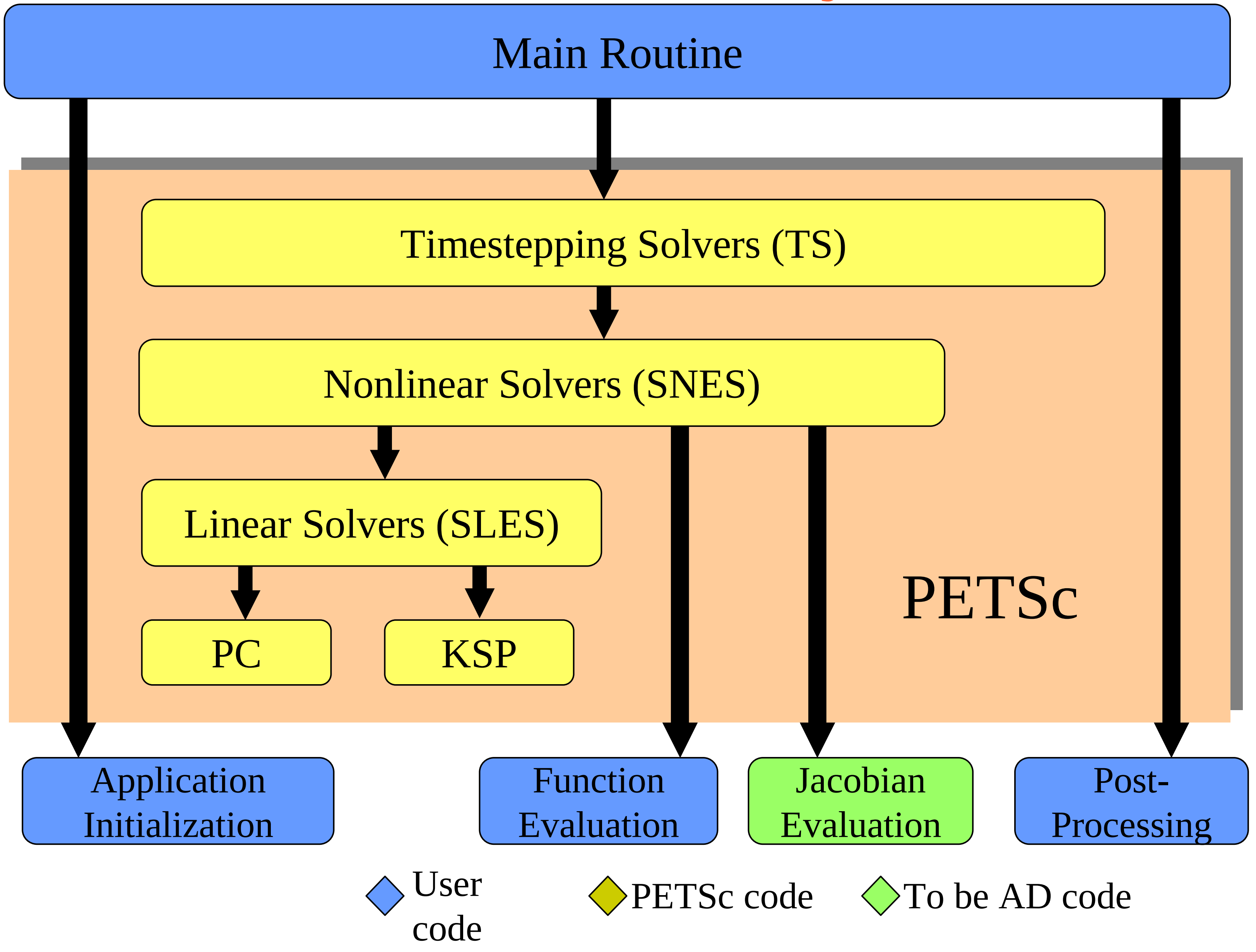
- **Developed under MICS at ANL to support research, prototyping, and production parallel solutions of operator equations in message-passing environments**
- **Distributed data structures as fundamental objects - index sets, vectors/gridfunctions, and matrices/arrays**
- **Iterative linear and nonlinear solvers, combinable modularly and recursively, and extensibly**
- **Portable, and callable from C, C++, Fortran**
- **Uniform high-level API, with multi-layered entry**
- **Aggressively optimized: copies minimized, communication aggregated and overlapped, caches and registers reused, memory chunks preallocated, inspector-executor model for repetitive tasks (e.g., gather/scatter)**

See **http://www.mcs.anl.gov/petsc**

User Code/PETSc Library Interactions



User Code/PETSc Library Interactions



Background of Hypre Library

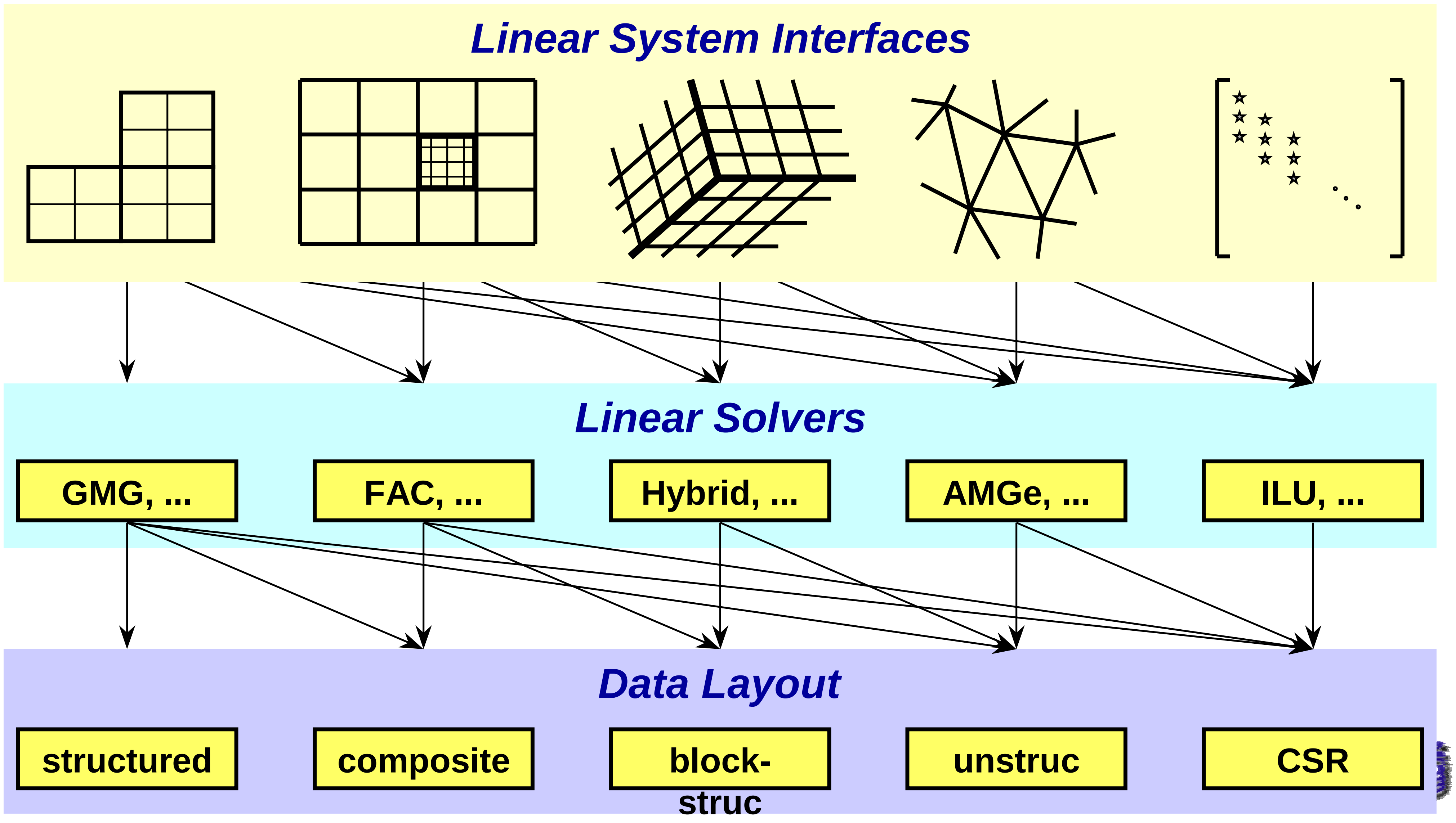
(to be combined with PETSc 3.0 under SciDAC by Fall’02)

- **Developed under ASCI at LLNL to support research, prototyping, and production parallel solutions of operator equations in message-passing environments**
- **Object-oriented design similar to PETSc**
- **Concentrates on linear problems only**
- **Richer in preconditioners than PETSc, with focus on algebraic multigrid**
- **Includes other preconditioners, including sparse approximate inverse (Parasails) and parallel ILU (Euclid)**

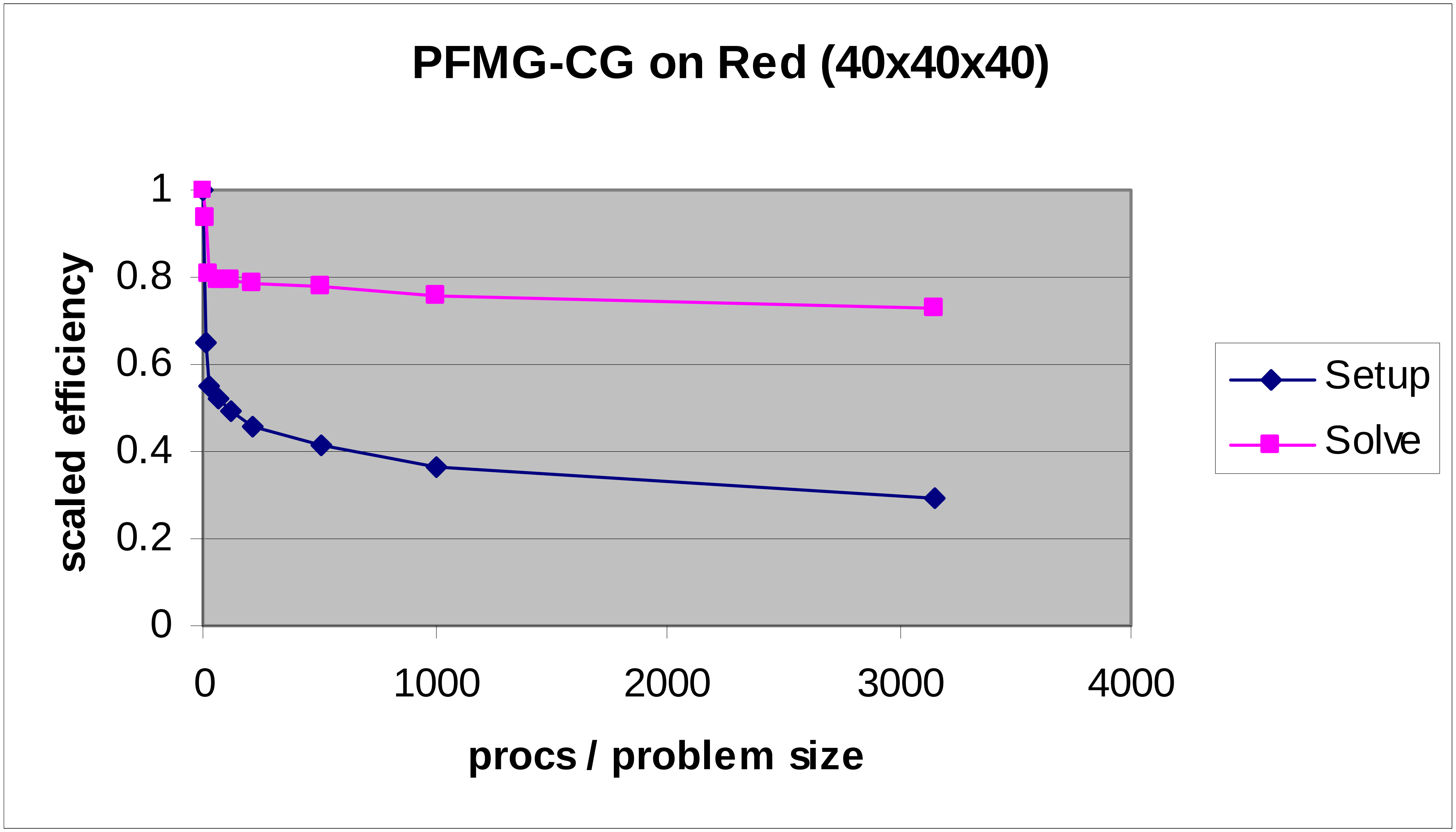
hypre
high performance preconditioners

See **http://www.llnl.gov/CASC/hypre/**

Hypre's "Conceptual Interfaces"



Sample of Hypre's Scaled Efficiency



Scope for TOPS

- **Design and implementation of “solvers”**
  - **Time integrators, with sens. analysis**

$$f(\dot{x},x,t,p)=0$$

  - **Nonlinear solvers, with sens. analysis**

$$F(x,p)=0$$

  - **Optimizers**

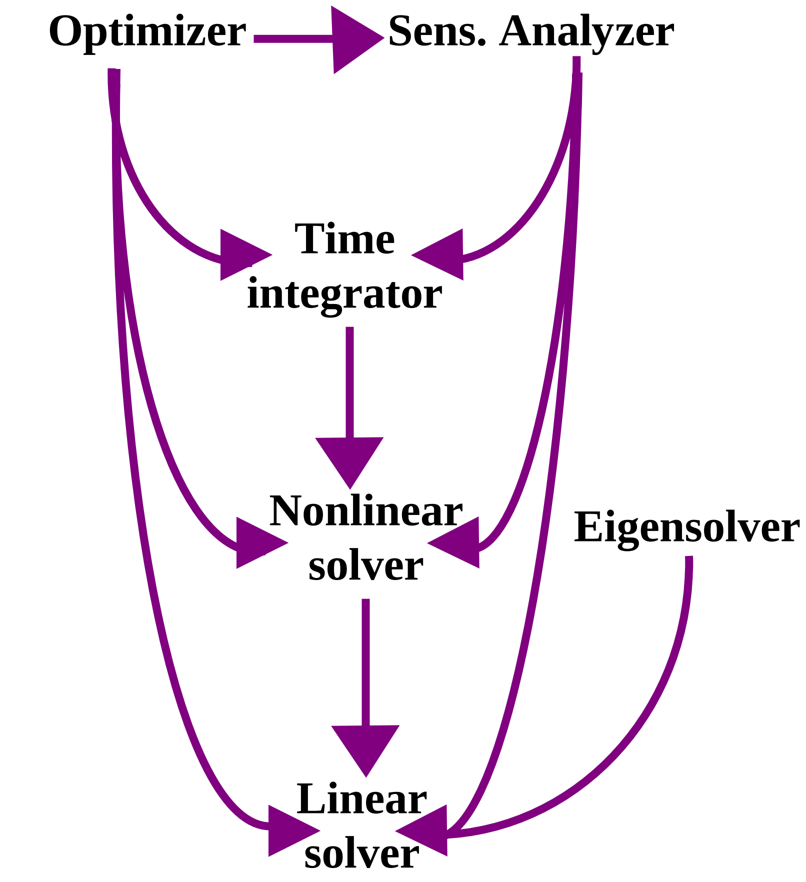
$$\min_{u}\phi(x,u)\ \ s.t.\ \ F(x,u)=0$$

  - **Linear solvers**

$$Ax=b$$

  - **Eigensolvers**

$$Ax=\lambda Bx$$



→ **Indicates dependence**

- **Software integration**
- **Performance optimization**

TOPS philosophy on PDEs

- Solution of a system of PDEs is rarely a goal in itself
  - **PDEs are typically solved to derive various outputs from specified inputs, e.g. lift-to-drag ratios from angles or attack**
  - **Actual goal is characterization of a response surface or a design or control strategy**
  - **Black box approaches may be *inefficient and insufficient***
  - **Together with analysis, sensitivities and stability are often desired**

$\Rightarrow$ Tools for PDE solution should also support related desires

TOPS philosophy on operators

- A continuous operator may appear in a discrete code in many different instances
  - Optimal algorithms tend to be hierarchical and nested iterative
  - Processor-scalable algorithms tend to be domain-decomposed and concurrent iterative
  - Majority of progress towards desired highly resolved, high fidelity result occurs through cost-effective low resolution, low fidelity parallel efficient stages

$\Rightarrow$Operator abstractions and recurrence must be supported

It’s 2002; do you know what your solver is up to?

- Has your solver not been updated in the past five years?
- Is your solver running at 1-10% of machine peak?
- Do you spend more time in your solver than in your physics?
- Is your discretization or model fidelity limited by the solver?
- Is your time stepping limited by stability?
- Are you running loops *around* your analysis code?
- Do you care how sensitive to parameters your results are?

If the answer to any of these questions is “yes”, you are a potential customer!

TOPS project goals/success metrics

TOPS will have succeeded if users —

- Understand range of algorithmic options and their tradeoffs (e.g., memory vs. time, inner iteration work vs. outer)
- Can try all reasonable options from different sources easily without recoding or extensive recompilation
- Know how their solvers are performing
- Spend more time in their physics than in their solvers
- Are intelligently driving solver research, and publishing joint papers with TOPS researchers
- Can simulate *truly new physics*, as solver limits are steadily pushed back (finer meshes, higher fidelity models, complex coupling, etc.)

Conclusions

- Domain decomposition and multilevel iteration the dominant paradigm in contemporary terascale PDE simulation
- Several freely available software toolkits exist, and successfully scale to thousands of tightly coupled processors for problems on quasi-static meshes
- *Concerted efforts underway* to make elements of these toolkits interoperate, and to allow expression of the best methods, which tend to be modular, hierarchical, recursive, and unfortunately — *adaptive*!
- Many challenges loom at the "next scale" of computation
- Undoubtedly, new theory/algorithms will be *part* of the interdisciplinary solution

Acknowledgments

- **Collaborators:**
  - **Xiao-Chuan Cai (Univ. Colorado, Boulder)**
  - **Dinesh Kaushik (ODU)**
  - **PETSc team at Argonne National Laboratory**
  - **hypre team at Lawrence Livermore National Laboratory**
- **Sponsors: DOE, NASA, NSF**
- **Computer Resources: LLNL, LANL, SNL, NERSC, SGI**

Related URLs

- Personal homepage: papers, talks, etc.
  - *http://www.math.odu.edu/~keyes*
- SciDAC initiative
  - *http://www.science.doe.gov/scidac*
- TOPS project
  - *http://www.math.odu.edu/~keyes/scidac*
- PETSc project
  - *http://www.mcs.anl.gov/petsc*
- Hypre project
  - *http://www.llnl.gov/CASC/hypre*
- ASCI platforms
  - *http://www.llnl.gov/asci/platforms*

Bibliography

- *Jacobian-Free Newton-Krylov Methods: Approaches and Applications*, Knoll & Keyes, 2002, to be submitted to J. Comp. Phys.
- *Nonlinearly Preconditioned Inexact Newton Algorithms*, Cai & Keyes, 2002, to appear in SIAM J. Sci. Comp.
- *High Performance Parallel Implicit CFD*, Gropp, Kaushik, Keyes & Smith, 2001, Parallel Computing 27:337-362
- *Four Horizons for Enhancing the Performance of Parallel Simulations based on Partial Differential Equations*, Keyes, 2000, Lect. Notes Comp. Sci., Springer, 1900:1-17
- *Globalized Newton-Krylov-Schwarz Algorithms and Software for Parallel CFD*, Gropp, Keyes, McInnes & Tidriri, 2000, Int. J. High Performance Computing Applications 14:102-136
- *Achieving High Sustained Performance in an Unstructured Mesh CFD Application*, Anderson, Gropp, Kaushik, Keyes & Smith, 1999, Proceedings of SC'99
- *Prospects for CFD on Petaflops Systems*, Keyes, Kaushik & Smith, 1999, in "Parallel Solution of Partial Differential Equations," Springer, pp. 247-278
- *How Scalable is Domain Decomposition in Practice?*, Keyes, 1998, in "Proceedings of the 11th Intl. Conf. on Domain Decomposition Methods," Domain Decomposition Press, pp. 286-297